SUPPORT SYSTEM FOR ON-BOARD CHECKOUT AND DATA MANAGEMENT SYSTEM (OCDMS), SATURN/APOLLO APPLICATION PROGRAM

PRC D-1177
Rev A

23 August 1967

Prepared for

National Aeronautics and Space Administration
George C. Marshall Space Flight Center
Huntsville, Alabama

PLANNING RESEARCH CORPORATION
LOS ANGELES, CALIFORNIA WASHINGTON, D.C.

SUPPORT SYSTEM FOR ON-BOARD CHECKOUT AND DATA MANAGEMENT SYSTEM (OCDMS), SATURN/APOLLO APPLICATION PROGRAM

PRC D-1177
Rev A

23 August 1967

Prepared for

National Aeronautics and Space Administration
George C. Marshall Space Flight Center
Huntsville, Alabama

Prepared by

J. L. Benson
D. H. Stomberg

PLANNING RESEARCH CORPORATION
LOS ANGELES, CALIF. WASHINGTON, D.C.

This document was prepared by Planning Research Corporation under Contract Number NAS8-20367, "An Airborne Evaluating Equipment Study," for the George C. Marshall Space Flight Center of the National Aeronautics and Space Administration. The work was administered under the technical direction of Quality and Reliability Assurance Laboratory, Marshall Space Flight Center, with Walter T. Mitchell acting as project manager.

Specification No. CG00N8-2036701A
Revision No. A
Release Date 23 August 1967

CONTRACT END ITEM DETAIL SPECIFICATION (COMPUTER PROGRAM)

PERFORMANCE/DESIGN AND PRODUCT CONFIGURATION REQUIREMENTS

(CEI 036701A)

SUPPORT SYSTEM FOR ON-BOARD CHECKOUT AND DATA MANAGEMENT SYSTEM (OCDMS), SATURN/APOLLO APPLICATION PROGRAM

Approved by C.H. Riley
(Planning Research Corp.)

Approved by _____

Date 23 August 1967

Approval Date _____

Contract Number NAS8-20367

PLANNING RESEARCH CORPORATION

SPECIFICATION ISSUE	ECP'S	PRODUCTION EFFECTIVITY
BASIC Date_____	ORIGINAL DESIGN ECP'S	*
  Date_____	INCORPORATED ECP'S	
  Date_____	INCORPORATED ECP'S	
  Date_____	INCORPORATED ECP'S	

* (Effectivity of the Specification on Production article or item Serial Number as Applicable).

(PLANNING RESEARCH CORPORATION) As Of 3/6/67

SPECIFICATION CHANGE LOG Superseding

SCN NO.	ECP NO.	SCN DATE	PAGES AFFECTED	ITEM EFFECTED

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF EXHIBITS

Specification No. CG00N8-2036701A
Revision No. A
Release Date 23 August 1967

CONTRACT END ITEM DETAIL SPECIFICATION (COMPUTER PROGRAM)

PART I

PERFORMANCE/DESIGN REQUIREMENTS

(CEI 036701A)

SUPPORT SYSTEM FOR ON-BOARD CHECKOUT AND DATA MANAGEMENT SYSTEM (OCDMS), SATURN/APOLLO APPLICATION PROGRAM

Approved by C.S. Riley
(Planning Research Corp.)

Date 23 August 1967

Contract Number NAS8-20367

Approved by _____

Approval Date _____

1.0 SCOPE

This part of this specification establishes the requirements for performance, design, test, and qualification of a group of computer programs identified as the Support System for On-Board Checkout and Data Management System (OCDMS), CEI 036701A. The Support System functions are language translation, data management, job management, and task management. It shall also provide the program production through the assimilation of the other functions and shall provide test and verification capabilities. This CPCEI provides the software tools and techniques to achieve the following major objectives:

a. Standardization of computer programming and data management practices to provide support of a spaceborne computer-based system for applications involving the Saturn/Apollo Application (S/AA) Experiment Program

b. Development of methods and procedures that provide for S/AA missions a traceability and correlation capability for all experimental data collected (that is, the OCDMS operational concept includes experiment integration of a space vehicle for prelaunch, in-flight, and post-flight activities)

c. Minimization of test planning cost and simplification of configuration management tasks by means of the control and information processing capability of the OCDMS software

d. Generation of S/AA experiment procedures and system software that provide maximum flexibility, and expandability by such means as the programming and engineering language features for the computer-based OCDMS

This CPCEI requires that all other elements and computer program components of OCDMS support software shall conform to, or be compatible with, the requirements delineated herein.

2.0 APPLICABLE DOCUMENTS

The following documents of exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between documents referenced here and the detailed contents of Sections 3, 4, and 10, the detailed requirements in Sections 3, 4, and 10 shall be considered superseding requirements.

2.1 Project Documents: None

2.2 Specifications

2.2.1 Performance and Design Requirements for the On-Board Checkout and Data Management System, General Specification for, Specification No. SS2036701A, June 1967

2.2.2 Performance/Design and Product Configuration Requirement, Supervisory System for On-Board Checkout and Data Management System (OCDMS) Saturn/Apollo Application Program, March 1967

2.3 Other Publications

2.3.1 MSFC-PROC-485, Input for Configuration Management Accounting and the Reporting System, Preparation of 28 October 1965

2.3.2 PRC D-1336, On-Board Checkout System Hardware Design, 11 November 1966

2.3.3 PRC D-850, On-Board Checkout System Software Design, 17 November 1966

2.3.4 SR-QUAL-65-48, NASA(MSFC), Directives for Software Development, 24 June 1966

2.3.5 Preparation of Contract End Item Detail Specifications (Computer Program) Exhibit XVIII to NPC 500-1, 26 May 1966

3.0 REQUIREMENTS

The OCDMS system requirements include performance requirements, design and construction requirements, and requirements for functional areas. These requirements define and control a spaceborne checkout and data management system to be used with the Saturn/Apollo Applications (S/AA) Experiment Program. Performance and design requirements included here are allocated from, identical with, or in recognition of requirements established by the system specification. (References 2.2.1 and 2.2.2)

3.1 Performance

OCDMS support system software performance includes performance requirements and requirements for functional areas. In general, this CPCEI is expected to provide functional operations which minimize on-line computation or data manipulation by virtue of the processing services.

3.1.1 System Requirements

The limits and/or capacities of this CPCEI performance shall be constrained to operational envelopes including program generation, data base processing and computer program verification/testing.

3.1.2 Operational Requirements

The Support System CPCEI is represented by the functional block diagram of Exhibit 1. Overall operational functions are identified in subsequent paragraphs. The identifications, descriptions, and relationships expressed for these CPCEI functions are intended for total systems operations and not as a restrictive design definition of computer program component (CPC) organization or as functional descriptions of particular main programs or subprograms.

3.1.2.1 Function 1: Language Translation

Language translation is a system function to be implemented by OCDMS Support System computer program processes and services. Machine translations and transliterations shall provide for source programming languages and data languages. Executable object code and parameters are to be produced in a form suitable for on-line operations under control of the OCDMS Supervisory System.

Attributes of this function are closely related to other identified system functions (i.e., 3.1.2.2 through 3.1.2.6). These relationships exist because the languages determine the degree to which actual communication and information acquisition is handled by the system. The

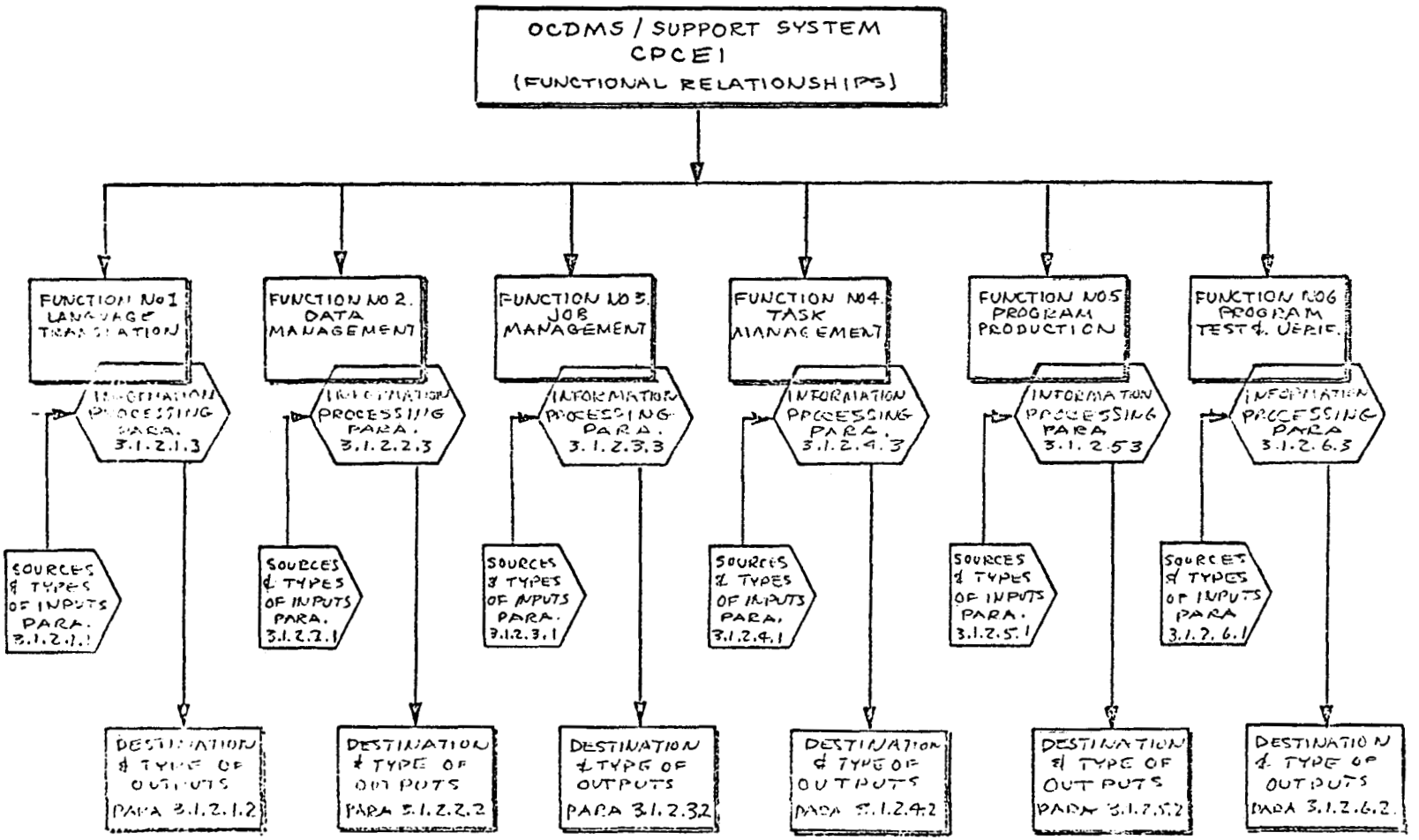


EXHIBIT 1 - OCDMS/SUPPORT SYSTEMS FUNCTIONS

other system functions are likewise involved in this mechanization of communication and information gathering processes and as such, contribute to salient language features including the following:

a. Program structure

b. Storage allocation

c. Data description

d. Data organization

e. Data conversion

f. Input/output control

g. Multitask operation

h. Listing processing

It is the intent for the purpose of this specification to define the language translation function as applying to all levels of programming or data language to be used by the OCDMS Support System. Thus, user-oriented languages or compiler languages for engineering and programming as well as assembly languages, and absolute machine code conversions are covered by the following operational requirements.

3.1.2.1.1 Source and Types of Inputs

a. Functional inputs shall typically be obtained from the following sources:

(1) Experiment procedures

(2) Equipment descriptions

(3) Equipment characteristics

(4) Experiment data definition

(5) Development test data

(6) Qualification test data

(7) Reliability test data

(8) Functional test data

(9) Calibration data

b. Units of measure (to be determined)

c. Limits/ranges (to be determined)

d. Accuracy/precision (to be determined)

e. Arrival frequency (to be determined)

3.1.2.1.2 Destination and Types of Outputs

a. Functional outputs shall consist of, but not necessarily be limited to, the following types:

(1) User-oriented language translation

(2) Assembly language processing

(3) Program structuring and semantics

(4) Data set descriptions and formats

(5) Data set conversion rules

b. Destinations (to be determined)

c. Units of measure (to be determined)

d. Limits/ranges (to be determined)

e. Accuracy/precision (to be determined)

f. Output frequency (to be determined)

3.1.2.1.3 Information Processing

Exhibit 2 summarizes the information processing requirements identified in the following subparagraphs.

3.1.2.1.3.1 Language Processing Criteria

The translation process shall be characterized by iterative operations; that is, it is expected that application programs will involve experiment control, equipment checkout, and self-check, and all other procedures will be submitted numerous times as jobs to the Support System. Thus, the process can be considered as an experimental method of successive approximations; the process terminating when the desired object code has been obtained.

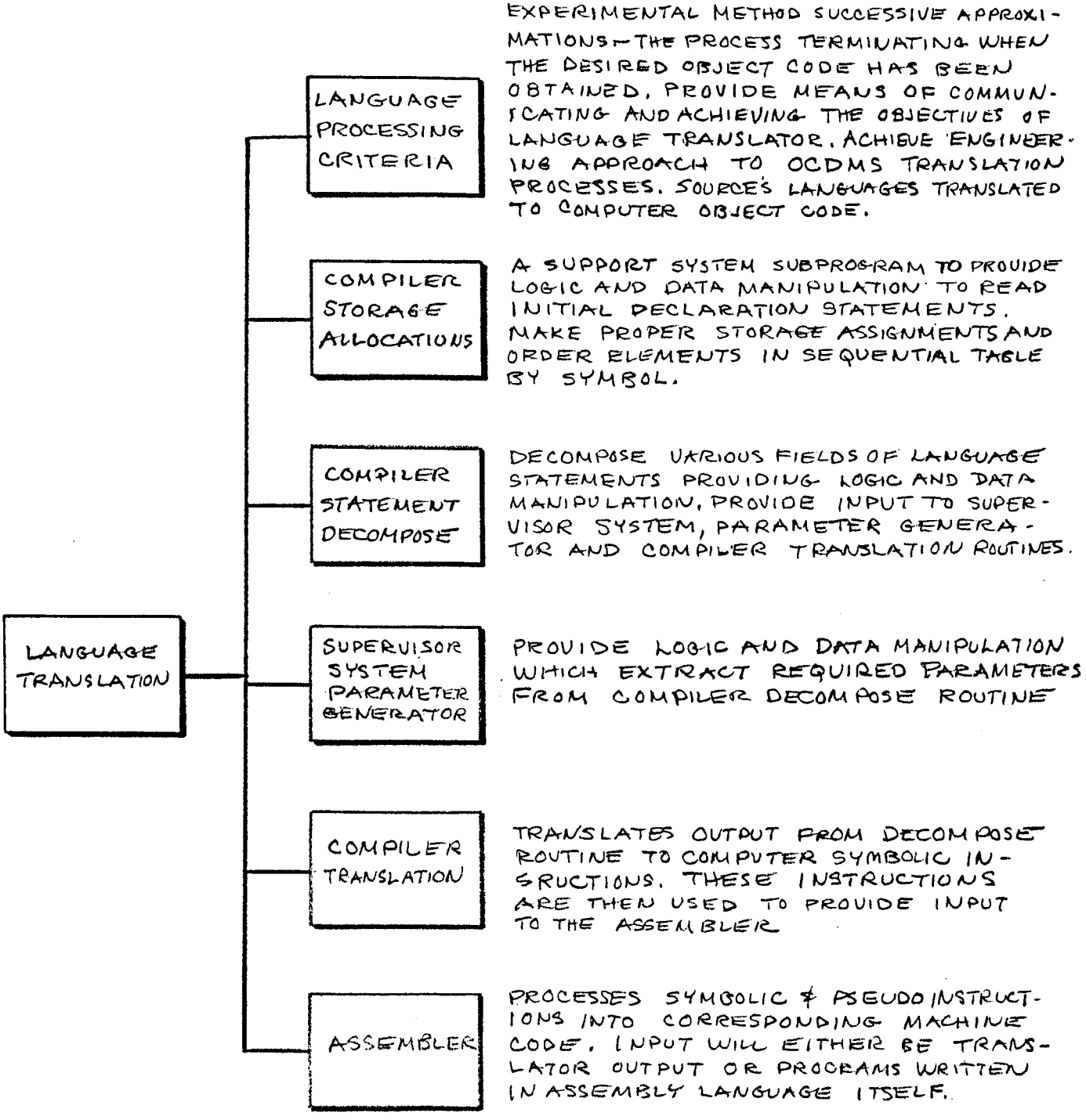


EXHIBIT 2 - LANGUAGE TRANSLATION INFORMATION PROCESSING

Each machine translation cycle in conjunction with program testing, elimination of errors, and more concise procedure statements shall provide the means of communicating and achieving the objectives of language translation.

Programming design that implements the language translation processes by brute-force techniques is not acceptable. This implies that the translation process shall not be limited to large tables of grammar rules, extensive dictionaries, and associated information processing restricted to table look-up, but shall be processed by subroutine call with the parameter as part of language modifier.

The principal design objective is to achieve an engineering approach to OCDMS machine translation processes--one which allows highly flexible, comprehensive communication and control to be exercised over the system. The method of implementation shall be by means of system macro-instructions by well-defined data management techniques, and by program testing services.

Source language statements shall in most cases be translated to computer object code in the form of linkages and calling sequences to functional subroutines developed for the OCDMS Supervisory System. Tables of parameters (linkage symbols, revolution information, system tape map, test symbol tables, device unit control blocks, scheduling algorithm table) will be generated as appropriate to enable OCDMS Supervisor routines to handle physical environment interactions, communication elements, and the resolution of possible conflicting demands. Source input that cannot be processed in the above manner shall be translated to machine instructions and appropriate operands. Linkage such as for macro-instruction or subroutine shall be generated for execution under Supervisor Control.

3.1.2.1.3.2 Compiler Storage Allocation

An OCDMS Support System subprogram shall provide logic and data manipulation to read the initial declaration statements and prerequisites specified by control statements. Proper storage assignments shall be made, and elements ordered in one sequential table by symbol. The declarations will include array-dimension information, labeled common declarations, equivalent statements, mode assignments, and the presetting of arrays. The double-entry table, ordered by the first entry (the symbol), would contain in the second entry a designation of the type of entry, the address assigned, and the mode.

3.1.2.1.3.3 Compiler Statement Decomposition

An OCDMS Support System subprogram shall provide logic and data manipulation to decompose various fields of the language statements. This is fundamentally a recursive process and as such imposes

large storage usage requirements because of normal postponement actions. It will therefore be expedient and necessary because of storage limitations to employ list processing techniques which postpone only the generation of connectives. The same reasons apply for the requirement to separate implementation of the expressions scanner and the statement scanner.

Statement decomposition shall be completely syntactical in execution, and should simply transform the statement to a discrete word array form. Such a form is regarded for the purposes of subsequent compiler subprograms processing as a modified list structure. The transformation would proceed on a basis that elements within each source statement would be isolated and entered in the symbol table. Addresses will replace symbols as they are encountered. Relative addresses will be generated for constants, statement labels, and function names as they are encountered.

3.1.2.1.3.4 Supervisor System Parameter Generator

An OCDMS Support System subprogram shall provide logic and data manipulation activities that extract required parameters of the Supervisor System from the array form described in the previous paragraph. The syntactical decomposition of a statement will transform the statements and repressions to discrete-word, symbol-free array form. The processing of the parameter generation algorithm will locate and move information required by the OCDMS Supervisory routines in performing operations associated with induced environment, scheduling parameters, communication conventions, priority levels, interruption actions, or queue supervision. The information will be accumulated in tabular form or as relocatable data sets contingent on the nature of the data. They will be positioned on the systems tape as a separate object module from the program statements in which originally specified. The symbolic location will be defined such that the Assembler, the Loader, or program production routines will provide the proper execution time linkage to all program segments that require access to the parameters.

3.1.2.1.3.5 Compiler Translation

An OCDMS Support System subprogram shall operate on the output of the compiler statement decomposition routines and translate the reduced and modified array form to OCDMS computer symbolic instructions. These computer instructions and macro-instructions must be subsequently assembled, relocated, and linkage must be generated for the object modules that are to be executed together. The translation algorithm would replace each line of the array as appropriate by a calling sequence that links and communicates parameters to functional subroutines of the on-line OCDMS Supervisor. In instances where a corresponding functional operating subroutine does not exist (that is,

where the array entry calls for a unique compilation sequence), the translation algorithm would replace the entry with directly executable machine language code. Translation shall not be accomplished immediately after the original statement scanning and storage allocation, for two reasons: (1) optimization can be performed more simply on the array form, and (2) it is easier to eliminate interregister redundancy when transforming the array to symbolic code. The output of the translator routine will be OCDMS assembly language symbolic code, relocatable addresses, and other address constants.

3.1.2.1.3.6 Assembler

An OCDMS Support System Assembly Program will process symbolic and pseudo instructions from the OCDMS computer repertoire into corresponding machine codes. Input will be either translator output or programs written in the assembly language itself. A wide range and variety of procedures and functional requirements are expected; thus, many subroutines will be written in assembly language. Minimum requirements, in addition to standard assembly practice, include:

a. Macro-instruction specifications and expansion

b. Repetition commands (such as a Duplicate Pseudo)

c. Output listing control

d. Segmentation and overlay

3.1.2.2 Function 2: Data Management

The data management function of this CPCEI shall provide the facilities for systematic and effective means of classifying, identifying, storing, cataloging, and retrieving all data (including loadable programs) processed by the Support System. The operational requirements shall be satisfied by CPC's which provide facilities grouped into two major categories:

- o Data Set Control
- o Data Access

a. Data Set Control

Data Set Control shall be provided by means of the following:

(1) Cataloging standards

(2) Storage location control

(3) File and program protection techniques

(4) Format standards

(5) Data set organization

b. Data Access

Input/output routines shall be provided that schedule and control the transfer of data between main storage and the input/output devices. Routines must be available to perform the following functions:

(1) Read data

(2) Write data

(3) Block and deblock records

(4) Overlap reading/writing and processing operations

(5) Read and verify data set labels

(6) Detect error conditions and correct them when possible

(7) Provide exits to appropriate error and label routines

3.1.2.2.1 Source and Type of Inputs

a. Functional inputs shall be obtained from the following sources:

(1) User language data sets

(2) Assembly language data sets

(3) Data set description

(4) Data description

(5) Data organization characteristics

(6) Input/output requirements

(7) System buffer requirements

b. Units of measure (to be determined)

c. Limits/ranges (to be determined)

d. Accuracy/precision (to be determined)

e. Arrival frequency (to be determined)

3.1.2.2.2 Destination and Types of Output

a. Functional outputs shall consist of, but not necessarily be limited to, the following:

(1) Data set organizations

(2) Data access control methods

(3) Block and buffering facilities

(4) Communication cell constructs

- Procedure Block Files
- Unit Control Blocks

b. Destinations (to be determined)

c. Units of measure (to be determined)

d. Limits/ranges (to be determined)

e. Accuracy/precision (to be determined)

f. Output frequency (to be determined)

3.1.2.2.3 Information Processing

Control and service routines shall provide processing facilities for system macro-instructions with functions to include the following:

a. Definition of control blocks for input/output operations

b. Buffer pool operations

c. Name specifications

d. Data access methods in which each input/output statement causes a corresponding machine input/output operation to occur

e. Data access methods that synchronize the transfer of data between programs using an input/output device and thereby eliminating delays for particular input/output operations

Exhibit 3 depicts this information processing.

3.1.2.3 Function 3: Job Management

The job management function of this CPCEI shall provide the facilities for efficient processing by:

a. Using the computing system to perform routine job handling activities in a rapid, precise manner

b. Eliminating nonessential operator decisions

c. Allowing the programmer to defer specification of input/output requirements until after compile time

Job control by programming personnel shall be accomplished at job set-up time by a job control language which will check for the required equipment to be on-line and functional and by the computer operator through system communication. System communication shall enable Support System to respond to operator commands, and to request that the operator perform such actions as mounting tapes; it must also permit a program to communicate with the operator.

Job scheduling may be implemented either as a sequential or priority scheduling system.

Provisions shall be made for options including:

a. Job accounting log

b. Concurrent job processing

c. Stacked job processing

3.1.2.3.1 Source and Types of Inputs

a. Functional inputs shall typically be obtained from the following sources:

(1) Unit Activity Block--one for each device in the system

(2) Buffer Activity Block--one for each buffer in the system

(3) Channel Activity Block--one for each channel in the system

(4) File Activity Block--one for each file in the system

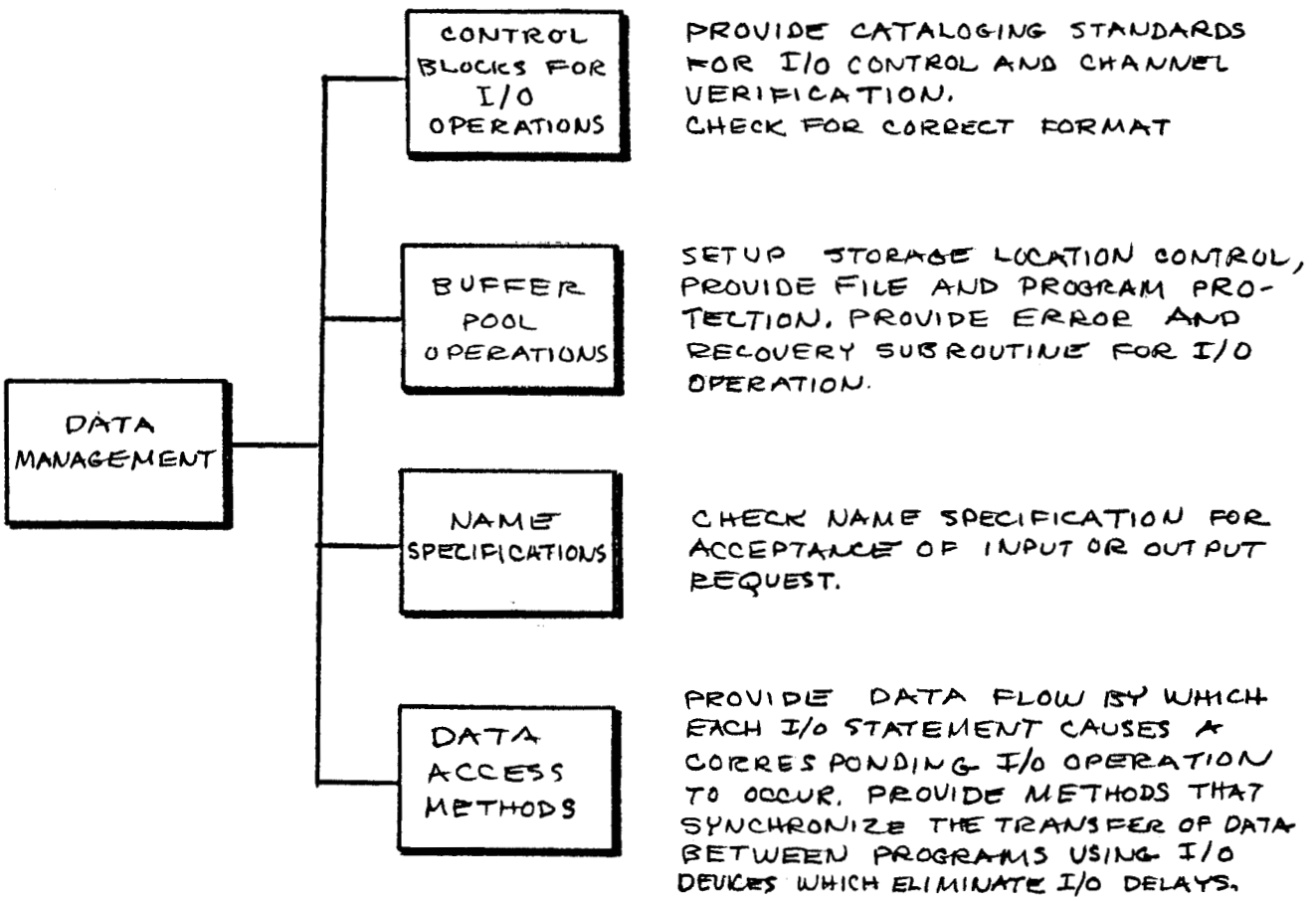


EXHIBIT 3 - DATA MANAGEMENT INFORMATION PROCESSING

(5) Pool Activity Block--one for each pool of buffers in the system

(6) Data handling requirements

b. Units of measure (to be determined)

c. Limits/ranges (to be determined)

d. Accuracy/precision (to be determined)

e. Arrival frequency (to be determined)

3.1.2.3.2 Destination and Types of Outputs

a. Functional outputs shall consist of, but not necessarily be limited to, the following types:

(1) Scheduling controls and parameters

(2) Data set dispositions

(3) Job file control blocks

(4) Control statements

(5) Accounting logs

(6) Label-processes

b. Destinations (to be determined)

c. Units of measure (to be determined)

d. Limits/ranges (to be determined)

e. Accuracy/precision (to be determined)

f. Output frequency (to be determined)

3.1.2.3.3 Information Processing

Control and service routines depicted by Exhibit 4 shall provide processing facilities for system macro-instructions with functions to include:

a. Simple program management

(1) Call a program

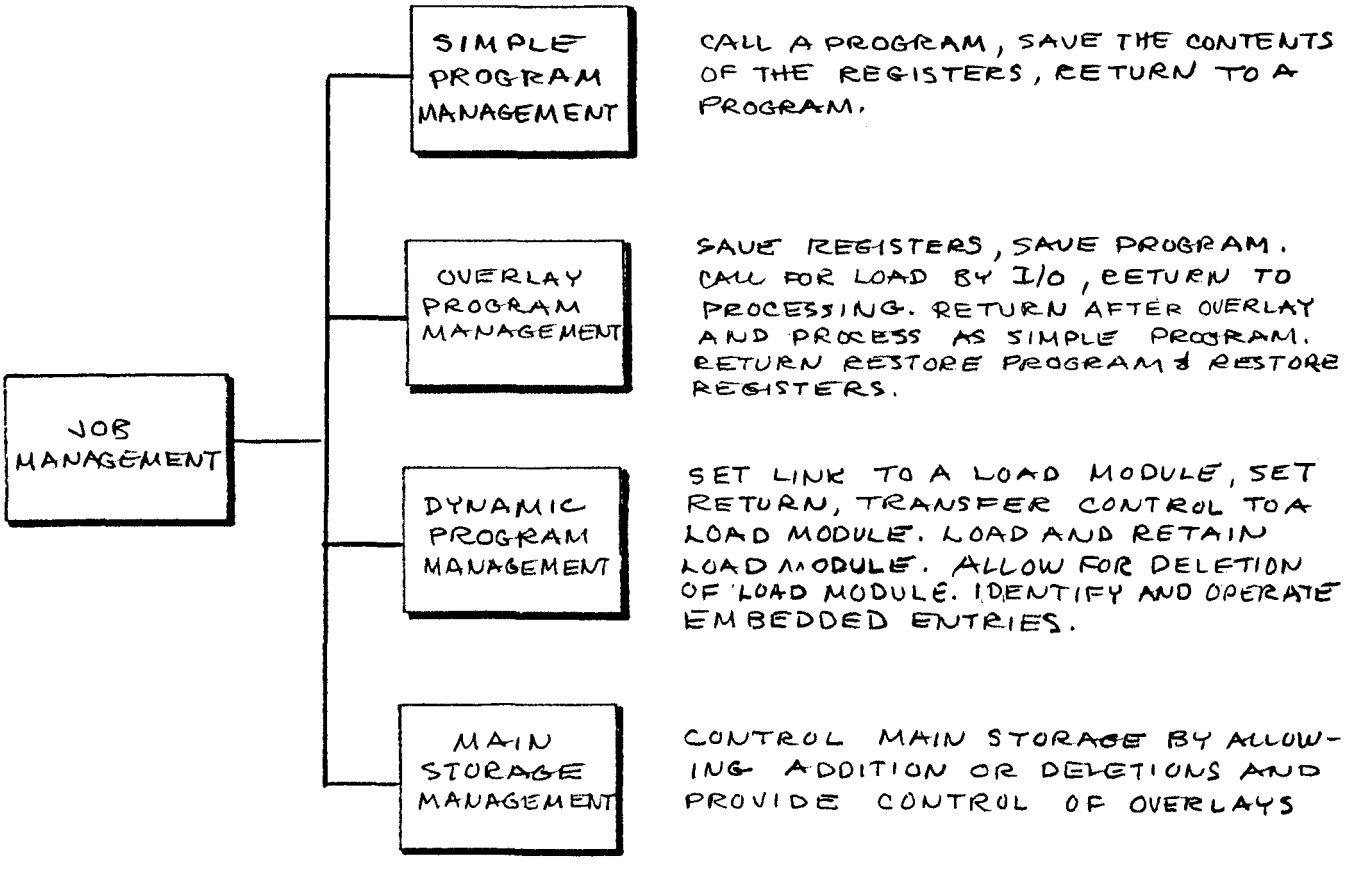


EXHIBIT 4 - JOB MANAGEMENT INFORMATION PROCESSING

(2) Save register contents

(3) Return to a program

b. Overlay program management

(1) Call and branch instructions

(2) Load segments while processing

(3) Load segments before processing

c. Dynamic program management

(1) Link to a load module

(2) Transfer control to a load module

(3) Load and retain load modules

(4) Delete retained load modules

(5) Identify embedded entries

d. Main storage management

(1) Allocate main storage

(2) Release allocated main storage

3.1.2.4 Function 4: Task Management

The task management function of this CPCEI shall be performed by control programs for units of work known as tasks. The performance of a task shall be requested by job steps. A distinction is made between a task and a program: programs are sequences of instructions, and tasks are the work to be done by executing programs.

The distinction is made for the purpose of providing multiprogramming capability, and therefore, the possibility of shared computer code. The same program shall in certain instances be used by more than one task.

Tasks, not programs, shall have priority for the multiprogramming considerations of this CPCEI.

Tasks in accordance with these definitions consist of computer program components that are executed under control of a task control block (TCB).

Switching of control from one task to another shall be accomplished with the contents of registers and the program status words of the relinquishing task stored in its task control block. When a task is dispatched, the task control block will contain all the necessary information to set up machine states required for task execution.

3.1.2.4.1 Source and Types of Inputs

a. Functional inputs shall typically be obtained from the following sources:

(1) Job file control blocks

(2) Scheduling controls and parameters

(3) Control statements

(4) Experiment operations

(5) Experiment data definitions

(6) Experiment displays

(7) Experiment controls

b. Units of measure (to be determined)

c. Limits/ranges (to be determined)

d. Accuracy/precision (to be determined)

e. Arrival frequency (to be determined)

3.1.2.4.2 Destination and Types of Outputs

a. Functional outputs shall consist of, but not necessarily be limited to, the following types:

(1) Single-task operations

(2) Multitask operations

(3) Storage allocation

(4) Task synchronization

(5) Exceptional condition handling

b. Destinations (to be determined)

c. Units of measure (to be determined)

d. Limits/ranges (to be determined)

e. Accuracy/precision (to be determined)

f. Output frequency (to be determined)

3.1.2.4.3 Information Processing

Control and service routines depicted by Exhibit 5 shall provide processing facilities for system macro-instructions with functions to include:

a. Task creation and management

(1) Create and attach a task

(2) Remove a task

(3) Change the dispatching priority

(4) Extract selected TCB fields

b. Task synchronization

(1) Wait for an event

(2) Signal event completion

c. Exceptional condition handling

(1) Specify program interruption exit

(2) Specify task abnormal exits

(3) Terminate tasks abnormally

(4) Checkpoint a job step

d. General services

(1) Request time and date

(2) Write to the operator

(3) Set/test timers

(4) Write to the log

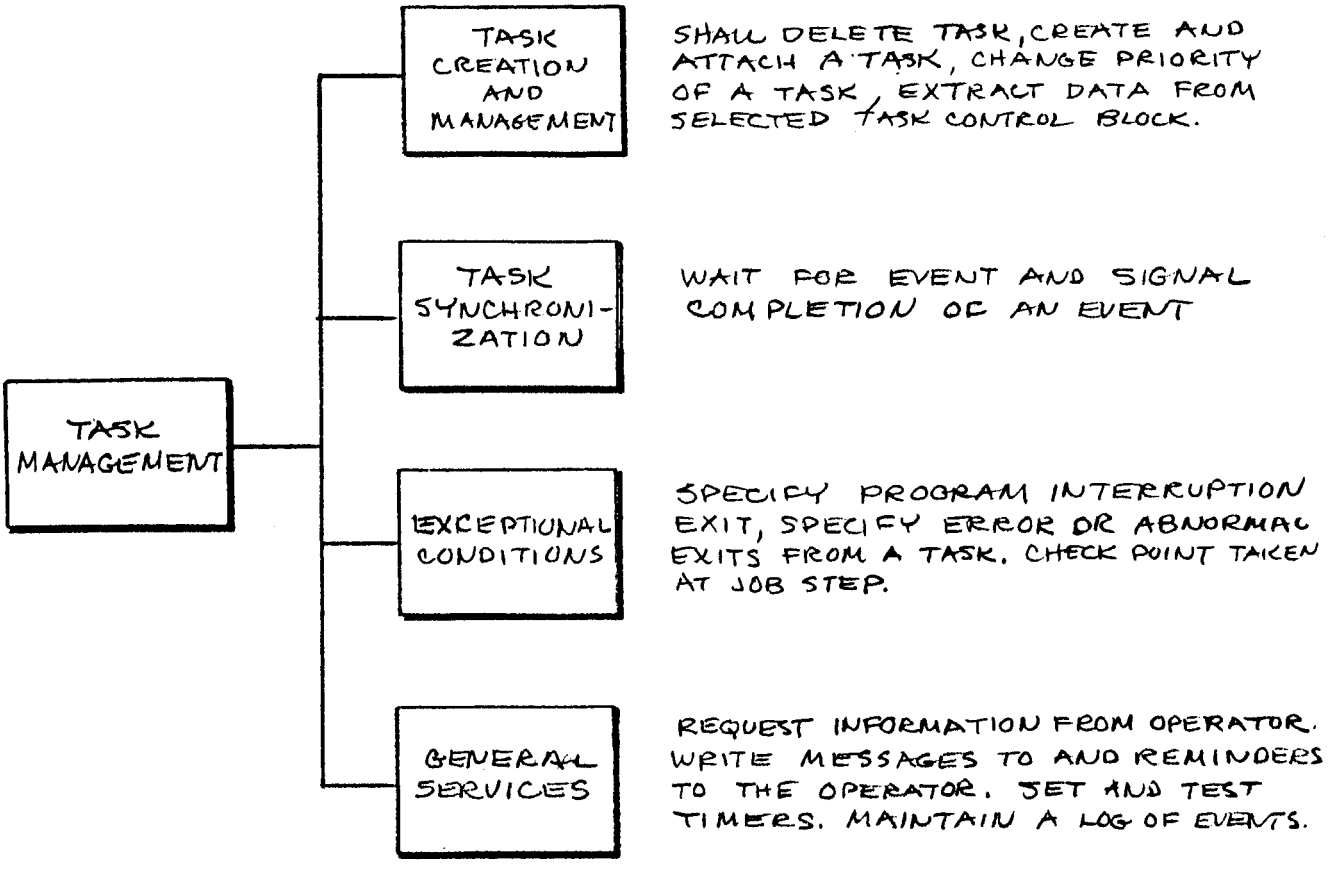


EXHIBIT 5 - TASK MANAGEMENT INFORMATION PROCESSING

3.1.2.5 Function 5: Program Production

The program production function of this CPCEI shall provide the processing capabilities to:

a. Provide the OCDMS supervisor with the control elements it requires, but are not limited to:

(1) Load module

- o Composed of all edited assembled and/or compiler modules
- o External symbol dictionary
- o Relocation dictionary
- o Testing symbol tables (when in test mode)

(2) Load module header file

- o Data set header label group
- o User header label group
- o Data set trailer label group

(3) Auxiliary storage directory

- o Define parameter location on tape
- o Location of subsystems files
- o Location of procedures files

(4) On-line system definition

- o Resident supervisor system
- o Linkage control

(5) Device unit control blocks

- o Auxiliary storage
- o CIU/signal adapters
- o CDU (display printer/keyboard)

- o PCM downlink
- o DCS uplink

(6) Procedure programs

- o Ordering of experiments or test
- o Alternate plan for experiment pretest problems

(7) Procedure schedule file

(8) Scheduling algorithm parameter table

- o Resolution of conflicting demands
- o Schedule program segment calls

b. Line separately assembled or compiled modules of programs into one load module

c. Incorporate modules from data sets other than its primary input, either automatically or upon request

d. Construct overlay programs for loading by a control program

e. Aid program modification by replacing, deleting, and rearranging control sections as directed by program production control statements

f. Reserve storage for the common control sections generated by the Assembler or other compiler languages

g. Provide processing options and logging diagnostic error messages

Program production processing shall be a necessary step that follows source program assembly or compilation. Load modules shall be processed for execution under control of the OCDMS Supervisory System CPCEI. Provisions for an error protection encoding scheme shall be required in conjunction with this system function (see Appendix 10).

3.1.2.5.1 Source and Types of Inputs

a. Functional inputs shall typically be obtained from the following sources:

(1) Translator object modules

(2) Data management CPC's

(3) Job management CPC's

(4) Task management CPC's

(5) Configuration management

- o Accounting requirements
- o Reporting system

b. Units of measure (to be determined)

c. Limits/ranges (to be determined)

d. Accuracy/precision (to be determined)

e. Arrival frequency (to be determined)

3.1.2.5.2 Destination and Types of Outputs

a. Functional outputs shall consist of, but not necessarily be limited to, the following types:

(1) Module linkage

(2) Library input sources

(3) Program overlay structure

(4) Program modifications and options

(5) Error protection encoding

(6) Data base edit/update

(7) Object to load module conversion

b. Destinations (to be determined)

c. Units of measure (to be determined)

d. Limits/ranges (to be determined)

e. Accuracy/precision (to be determined)

f. Output frequency (to be determined)

3.1.2.5.3 Information Processing

Exhibit 6 depicts the information processing requirements that shall include:

a. Module Linkage: Processing shall make it possible to segment programs into one or more control sections. Each module may be separately assembled or compiled. The program production routines provide the facilities for combining such modules into a single-load module.

b. Additional Inputs: Standard subroutines from the OCDMS program library may be added to an output module. Symbols that are undefined after all modules have been processed in a particular processing run shall call an automatic library call mechanism. This will cause a library search to resolve the reference. A named module that is found shall be processed to be an integral part of an output load module.

c. Program Modification: By means of replacement, deletion, or movement as directed by control statements shall be provided. External symbols shall be changed or deleted as directed by other control statements to the program production routines.

d. Options and Error Messages: Shall be processed by routines to produce maps or a cross-reference table which show arrangement of control sections and also indicate how communication is to be accomplished. Special processing options that negate automatic library call or the effect of minor errors shall be provided. Throughout, processing errors and possible error conditions must be logged.

3.1.2.6 Function 6: Program Test and Verification

The program test and verification function of this CPCEI shall include provisions for execution-time testing of input programs. The test services to be performed shall be specified by the programmer through macro-instructions included in the source program.

Services shall be performed at specified points in the program under test. Macro-instructions and application program instructions may be intermixed, grouped separately, or even assembled independently.

Service routines shall be reenterable and usable by several tasks in multiprogrammed tests. These routines require protection against modification by application programs, and shall include facilities to detect runaway testing and excessive test output.

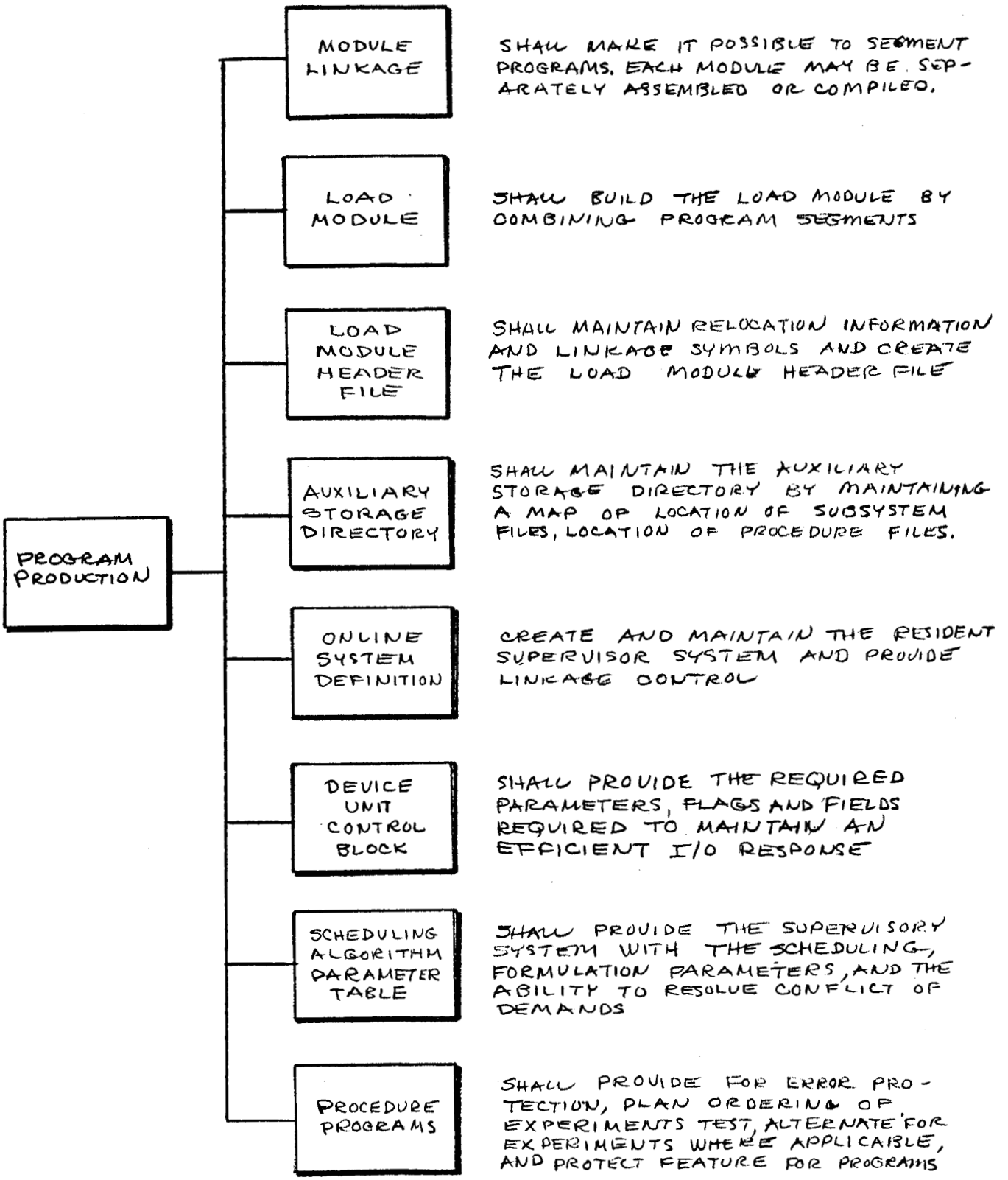


EXHIBIT 6 - PROGRAM PRODUCTION INFORMATION PROCESSING

The test data shall be processed by an edit routine that provides options in recording and printing according to output selection codes expressed in the related macro-instructions. The test data, including all associated symbols, shall be printed in a format defined in the source program.

3.1.2.6.1 Source and Types of Inputs

a. Functional inputs shall typically be obtained from the following sources:

(1) Translator/assembler object modules

(2) Program production modules

(3) Implementation test requirements (see Paragraph 4.2)

b. Units of measure (to be determined)

c. Limits/ranges (to be determined)

d. Accuracy/precision (to be determined)

e. Arrival frequency (to be determined)

3.1.2.6.2 Destination and Types of Outputs

a. Functional outputs shall consist of, but not necessarily be limited to, the following types:

(1) Dump or record processes

(2) Trace actions

(3) Test identifiers

(4) Branch and jump instructions

(5) Initialization and set-up

b. Destinations (to be determined)

c. Units of measure (to be determined)

d. Limits/ranges (to be determined)

e. Accuracy/precision (to be determined)

f. Output frequency (to be determined)

3.1.2.6.3 Information Processing

Control and service routines depicted by Exhibit 7 shall provide processing facilities for system macro-instructions with functions to include the following:

a. Recording of main storage, identified changes, storage maps, system tables, and comments

b. Tracing of program transfers, call statement executions, and storage references

c. Test actions to initiate testing, to perform test at location, to define flags or counters, to alter sequences or conditions, and to terminate testing

d. Subroutine and program transfers

e. Assignment of conditions and values

3.1.3 Data Base Requirements

Parameters which affect the design of the OCDMS CPCEI will reflect information in the following categories:

a. Calibration Information: Coefficients of curves determined during calibration of transducers or other measuring devices against established measurement standards.

b. Configuration Information: Part numbers, serial numbers, and other identifiers that establish the particular configuration of critical or otherwise important components and physical assemblies.

c. Experiment Monitor Tolerance Values: Magnitudes of upper and lower tolerance limits to nominal values of measured quantities.

d. Site Adaption Parameters: Conversion and format variables used to transliterate alphanumeric data for site peculiar and multisite computer differences. (See Paragraph 3.2.1 for additional identification.)

e. Reference Designator and Device Codes: Symbolic labels assigned to hardware components, assemblies, or subsystems that provide a convenient cross-reference between different engineering documentation.

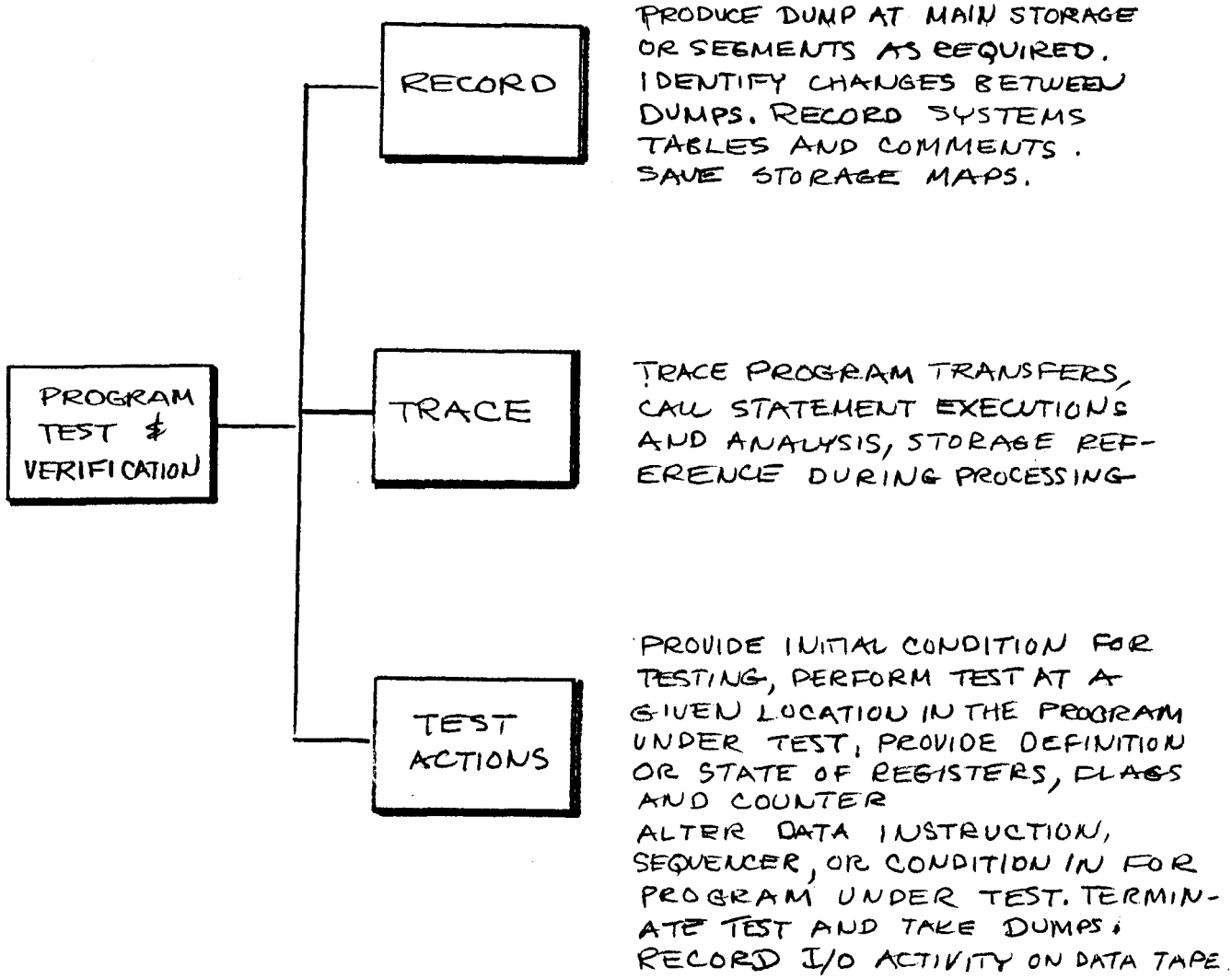


EXHIBIT 7 - PROGRAM PRODUCTION INFORMATION PROCESSING

f. Measurement Numbers and Names: Primarily this is a reference to PCM telemetry time slots, but also appropriate for all measurement signal paths.

g. System Addresses and Identifiers: Binary bit patterns that are assigned to enable communication to multiple ground site stations, to other modules, or to major subsystems of the OCDMS.

h. Test Point Dictionary: OCDMS and experiment equipment test point addresses, mnemonic names, and other convenient information associated with test points.

i. Data/Number Conversion Tables: Cross-reference lists between BCD, ASCII, EBCDIC, decimal, octal, hexadecimal, floating-point, binary, and numerous other conversions encountered in the normal course of computing (particularly for multicomputer applications).

j. Control Words and Masks: Standard bit patterns used to set up devices such as programmable function generators, multiple discrete generators, and similar equipment.

3.1.4 Human Performance

In accordance with requirements identified by Reference 2.1.2, General Specification for the OCDMS, the CPCEI shall have performance characteristics that reflect established human engineering design standards. In order to enhance the reliability of human performance, and to reduce operating inefficiencies, and training requirements, MSFC-STD-267A, Human Engineering Design Criteria, shall be used as a guideline for OCDMS design, and software development shall be in accord as applicable.

The computer software design shall not impose a minimum time for human design making or response. It shall respond to external commands in a maximum time of 1 second. Emergency-type messages shall be displayed in a blinking mode of operation.

3.2 CPCEI Definition

The functional relationship of the CPCEI to other equipment/computer programs and the identification of Government-furnished computer programs incorporated in the CPCEI are specified by the following subparagraphs.

3.2.1 Interface Requirements

The OCDMS Support System CPCEI will be utilized by programming, engineering, and computer technician personnel. The overall

system or segments of it will be used for mission support activities relating to various installations, sites, and operational locations as specified by the OCDMS General Specification. These factors are the basis for interface requirements identified below.

3.2.1.1 Interface Block Diagrams

Exhibits 8, 9, 10, and 11 portray the relationships to other equipment/computer programs with which this CPCEI shall interface.

3.2.1.2 Detailed Interface Definition

3.2.1.2.1 OCDMS Computer (to be determined)

3.2.1.2.2 Computer Language

The language translation function requirements identified in 3.1.2.1 and subparagraphs shall be designated also as interface requirements.

3.2.1.2.3 Data Management

The data management function requirements identified in 3.1.2.2 and subparagraphs shall also be designated as interface requirements. The reason for this definition is based on the system interactions that result because of data organization, descriptions, and formats. The generation of site-peculiar parameters by the OCDMS Support System is a significant requirement in this respect (3.1.2.1.3.4 and 3.1.2). Adaptability and flexible interface relationships are to be design goals that must be achieved with minimum impact to existing facilities and operational procedures.

3.2.1.2.4 OCDMS Supervisor System CPCEI

Reference 2.2.2 specifies performance/design characteristics which shall be designated interface requirements.

3.2.1.2.5 OCDMS Experiment Procedure CPCEI (to be determined)

3.2.2 Government-Furnished Property List (to be determined)

3.3 Design Requirements

This section of the specification contains requirements and standards that affect the design of the CPCEI and are distinguishable from the performance requirements of Section 3.1.

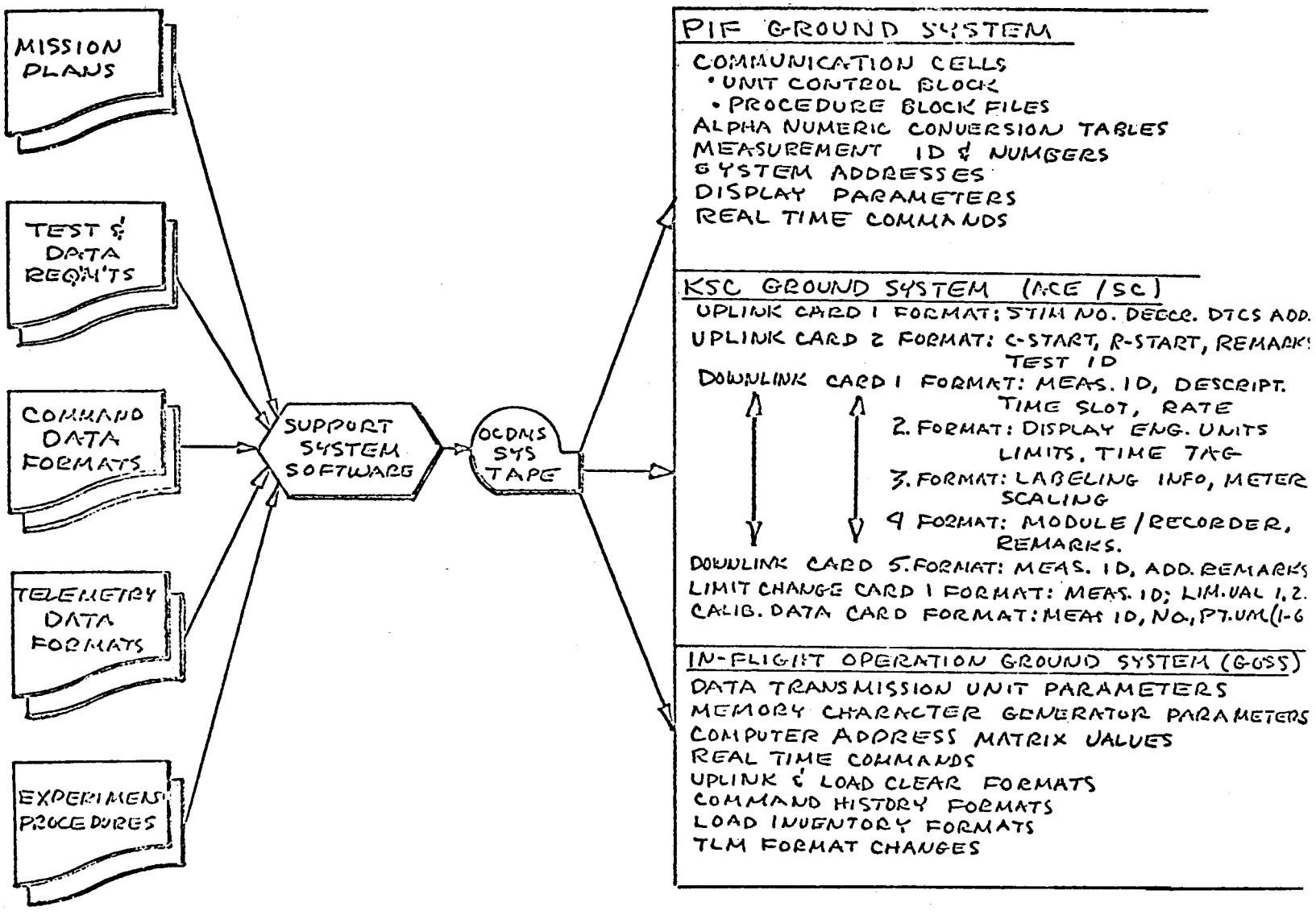
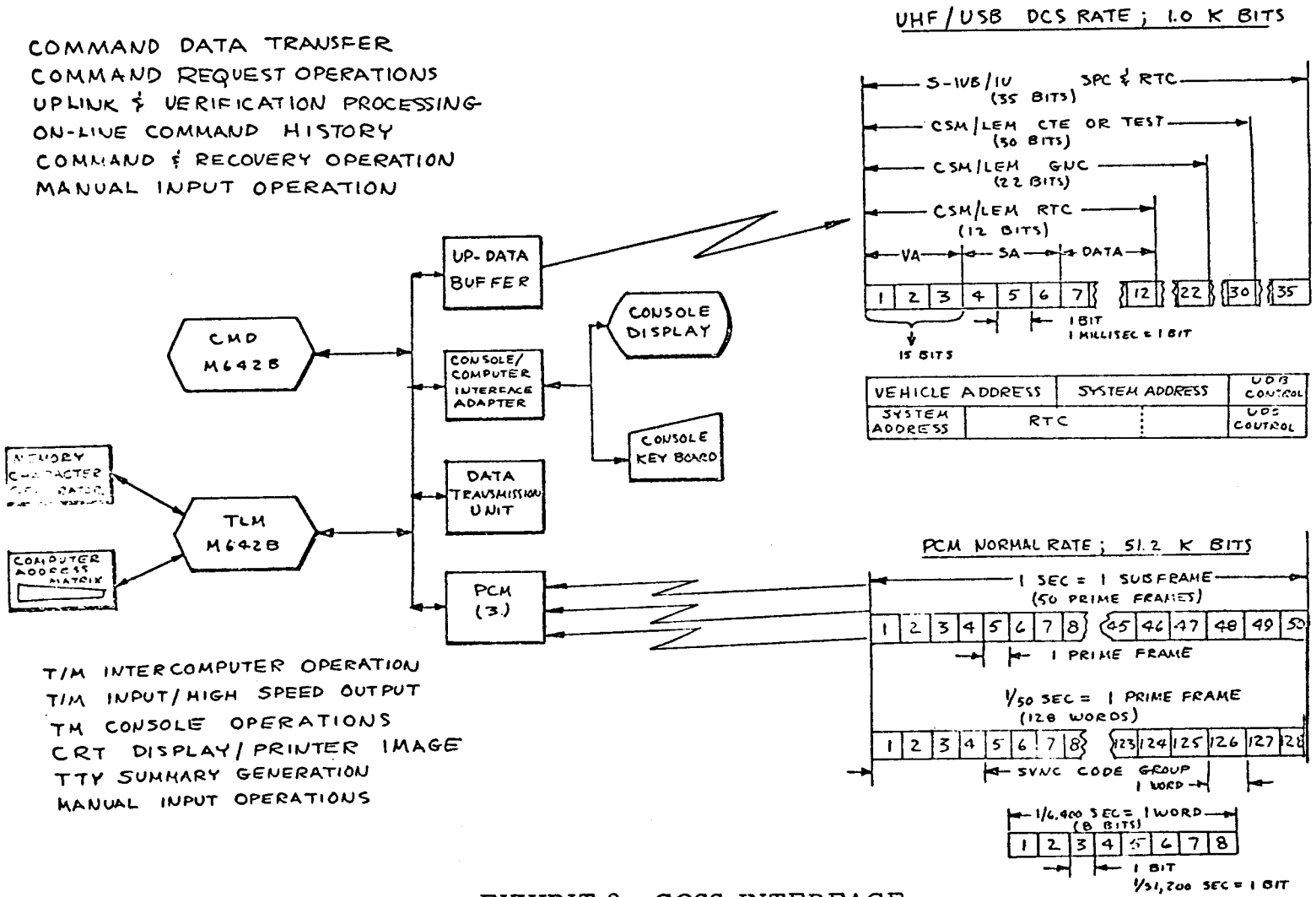


EXHIBIT 8 - OCDMS SITE-PECULIAR PARAMETERS

COMMAND DATA TRANSFER
COMMAND REQUEST OPERATIONS
UPLINK & VERIFICATION PROCESSING
ON-LINE COMMAND HISTORY
COMMAND & RECOVERY OPERATION
MANUAL INPUT OPERATION



T/M INTERCOMPUTER OPERATION
T/M INPUT/HIGH SPEED OUTPUT
TM CONSOLE OPERATIONS
CRT DISPLAY/PRINTER IMAGE
TTY SUMMARY GENERATION
MANUAL INPUT OPERATIONS

EXHIBIT 9 - GOSS INTERFACE

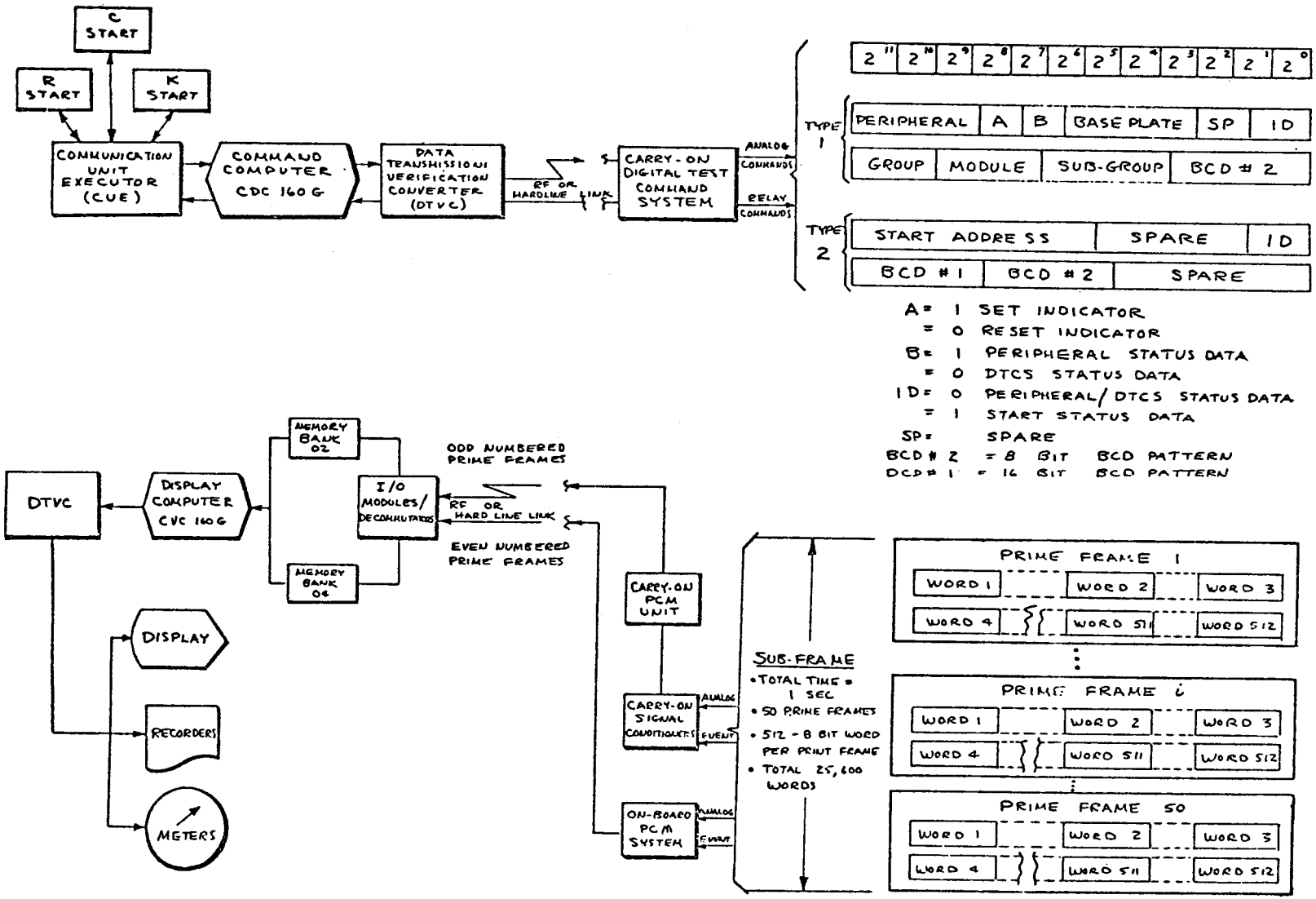


EXHIBIT 10 - ACE/SC INTERFACE

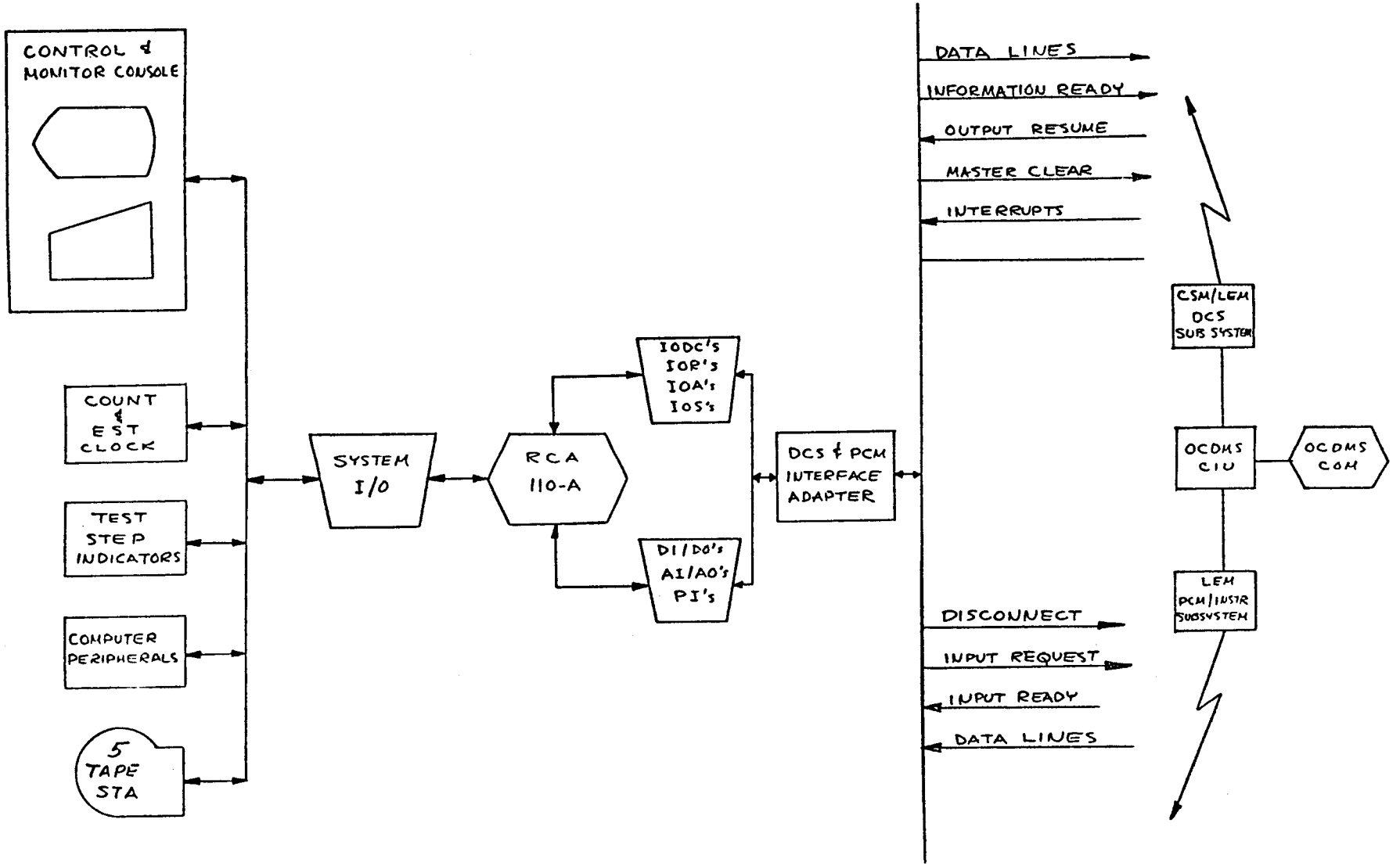


EXHIBIT 11 - PIF INTERFACE

3.3.1 Programming Standards

The use of programming standards and conventions assures compatibility between computer program components. Since a system development of the scope and complexity of OCDMS involves a large number of different contributors, engineering disciplines, and programming methods, then adherence to these standards is mandatory in order to avoid significant and complicated communications problems.

3.3.1.1 General Requirements

Ensure intended program initialization. Avoid assumptions that storage is initially zero, tapes are positioned, etc.

Parameterize where possible. Most numbers should be assembly parameters defined in one region and referred to symbolically throughout a program. Even numbers which are absolute in context, such as positions in a table or in a calling sequence, should be symbolic so that all references to the table or calling sequence can be found in a cross-reference dictionary.

In testing a series of conditions, as a parameter with several possible states, use positive testing of each condition or state. Do not assume that the set of mutually exclusive conditions in a functional sense will exhaust the set of conditions that may physically exist. For example, if it is known that a field may legitimately contain a code of A, B, or C, all three conditions should be tested explicitly rather than assume it is C because it is not A or B. If none of the conditions are satisfied, an exit to an unusual end routine should be made.

3.3.1.2 Commentary

a. Comments: Each routine should begin with a series of "comments" cards containing the following: routine label; routine name, if any; main program and subprogram label in whose context this routine is executed; function or purpose; calling sequence; entry conditions, exit conditions; error exits; and labels of the other routines or data blocks used. Use comments and blank spaces (blank comments cards) to show program structure and sequence.

b. Remarks: Symbolic coding lists should make liberal use of the "remarks" field. They should be meaningful and concise. Avoid the type of remark which merely repeats that which is obvious from the coding. Remarks should be included to denote program entry points, loop exit points, loop return points, program exit points, major "block" begin and end, and modified instructions and switches.

3.3.1.3 Coding

Every label appearing on a flowchart must appear in the coding.

Symbolic relative addressing should be used only when limitations on the total number of labels requires such use to save labels.

Chosen techniques must be implemented in as comprehensible form as possible (i.e., coding should be straightforward, non-"tricky"). Where efficiency requirements preclude this, commentary must fully explain the coding.

All storage areas should be defined:

a. Instructions

b. Fixed constants

c. Variable constants

d. Working storage

e. Input/output buffer areas

3.3.1.4 Communication Practices

Routines should be entered by calling sequences specified by the appropriate operating system, library, or user group.

Routines may have one or more entries, where each entry performs a single, well-defined function. Multiple-entry regions are written to share coding or data used by related functions.

Standard subroutine entry and exit macros are employed by all regions except where a programmer-defined exit is necessary to facilitate multiple returns from routines.

Routines are generally entered from a "control" routine and return to that routine at the conclusion of execution. All programs written for execution when the interrupt system of the computer is enabled should give due regard to ensuring that program context is preserved in the event of the occurrence of an interrupt.

3.3.2 Program Design

Four levels of program specification are involved in the OCDMS Support System design: (1) main program, (2) subprogram, (3) routines and data sets, and (4) regions.

3.3.2.1 Organization Guidelines

3.3.2.1.1 Main Program and Subprogram

a. Main Program: A main program is defined as a logical processing entity which accomplishes a major system function. It occupies the highest level in the hierarchical structure of a program, and may be coded as an open routine. An example is a single pass of the OCDMS multipass compiler.

b. Subprogram: A subprogram is the first level of division in the hierarchy of a main program, accomplishing a major task or group of subtasks of the main program. It is coded as a routine and may use one or more routines and/or data sets.

3.3.2.1.2 Routines and Data Sets

a. Routines: The conventional definition of a subroutine is to be used. It may be described as a closed sequence of machine instructions with entry and exit points permitting it to be executed at arbitrary locations within higher level programs, and which performs a distinct function, task, or module of processing.

(1) Basic Routine: A basic routine uses no subroutine.

(2) Standards for Routine Design

- The structure of a routine shall be such that it may be specified, flowcharted, coded, and checked out as an independent entity, given that subroutines linkages and executions are proper.
- Generally, the size of a routine shall not exceed 100 to 200 instructions. No minimum size requirement exists.
- A routine shall not contain logic or structure related to more than one functional task. Subsequent program modifications are significantly simplified with routines of greater functional unity and less scope.
- Whenever possible, a routine shall maintain a given hierarchy level throughout a particular usage; at the highest level (e.g., a subprogram), routines tend to become characterized as a sequence of subroutine linkages and calling sequences.
- Routines (at any level) must not refer to locations within other routines. Routines may communicate through separate cells, data sets, or tables in data regions.

- o Routines must ensure that the contents of all operating registers not named in their calling sequence or usage conventions are unchanged at exit from conditions at entry.

b. Data Sets: A data set is the major unit of data storage and retrieval in the OCDMS Support System. It consists of a collection of data in one of several prescribed arrangements and described by control information that the system has access to.

(1) Data Set Identification: A data set is a named, organized collection of one or more records that are logically related. Information in the data set is not restricted to a specific type, purpose, or storage medium. A data set may be, for example, a source program, a library of macro-instructions, or a file of data records processed by an application program.

(2) Standards for Data Sets

- o A data set shall reflect unity of function in logical content and organization.
- o A data set shall occupy sequentially contiguous sections of storage except as limited by a particular input/output device.

3.3.2.1.3 Regions

A region is defined as an aggregate of routines (a program region) or an aggregate of data sets (a data region). A program region is not itself an executable routine, but a collection of routines which must be categorized as serving the same broad functional purpose. For example, one region may consist of all error message routines; another region may contain only printing routines. A data region may consist of only one data set, or may contain several related tables or a generation data group. A generation data group is a collection of successive, historically related data sets.

In general, a program region should correspond approximately with a certain hierarchy level in the program structure. A generation data group should be present in main storage only during such operations as edit and update of the data base.

3.3.2.2 Subprogram Construction

The Support System shall provide provisions to combine subprograms of all OCDMS CPCEI's at four distinct times during the cycle from program statement to complete job execution.

3.3.2.2.1 Compilation/Assembly Time

Subroutines and separately written source decks may be combined as the input to a single compilation or assembly.

3.3.2.2.2 Program Production Time

Separately compiled object modules and load modules can be included as input to a single program production run to produce a single composite load module for execution. If overlay techniques are used, the entire load module need not be contained in available main storage.

3.3.2.2.3 Task Performance Time

A load module may be interconnected with other named modules during the time the task is performed. A job control statement (EXECUTE) may designate a job step by identifying other named modules to be fetched and executed. In general, the first load module used in the performance of any task may interconnect with other load modules named during the time the task is performed.

Modules shall be placed wherever storage is available, relocated (i.e., initialized to execute from a chosen location), and interconnected dynamically. Overlay of entire load modules shall at times be shared between different tasks in a multitask operating environment.

3.3.2.2.4 Program Specified Time

The same sequence of instructions shall (as appropriate to its specific status of three different possible forms; i.e., source, object, or load module) be capable of being used as a subprogram in a larger sequence of instructions. This usage shall be possible with little or no change at any particular specified time.

3.3.2.3 Data Communication

The OCDMS Support System shall be governed by the following conventions for data communication between subprograms.

3.3.2.3.1 Implicit Communication

More than one subprogram can know the location of data and have access to it because they were compiled, assembled, or serviced by program production routines together.

Usage of implicit communication shall be possible when subprogram linkage is either branch and link instructions or CALL-type macroinstructions. Subprograms that require modification and as a result have

changed working or data storage will normally be required to be re-written with explicit communication.

3.3.2.3.2 Explicit Communication

Data communication between programs shall be possible at the time of linkage. Either the data or its location may be passed between the programs. Communication by passing the data itself is called communication by value; communication by specifying the location of the data is called communication by name.

Data communicated explicitly are called parameters. Parameters are further classified according to their use, as follows:

a. Control program parameters, which are passed between the application programs and the control program when system macro-instructions are executed. These parameters are transferred in either parameter registers or parameter lists.

b. Application program parameters, which are transferred between subprograms of the experiment procedure programs when linkages are defined. These parameters are transferred only by parameter lists.

3.3.2.4 Program Control

Program control shall change in an upward or downward sense within the hierarchy of the CPCEI, depending on each program linkage. An exception to the up/down change is possible in the case of certain system macro-instructions in which the given control level is kept constant (i.e., a transfer to another module at the same level). The allowable linkage types are specified by the following paragraphs.

3.3.2.4.1 Direct Linkage

Direct linkage is established by a branch and link instruction or a system macro-instruction of the CALL type. This type of linkage is used to link two subprograms or an application program to a supervisory or control routine. (Most system macro-instructions for the data management function would use this type.)

3.3.2.4.2 Support Program-Assisted Linkage

Support System macro-instructions of the type used for dynamic program management or task creation and management shall be used to link two subprograms.

3.3.2.4.3 Supervisor Linkage

Control program routines that are executed in a supervisory state shall be provided for by a nonambiguous linkage instruction.

3.3.2.4.4 Exit Linkage

Synchronous exits (occurring concurrently and with a regular or predictable time relationship) shall be provided in response to system macro-instructions to enter application routines from a system service program.

Asynchronous exit (unexpected or unpredictable with respect to instruction sequence) shall be allowed by use of a load program status word for exit linkage and entry to an application program.

3.3.2.5 Program Element Names and Labels

a. Formats: The standard format for data set and subprogram element names shall be one to eight characters, starting with an alphabetic character, without embedded blanks. The special character period (.) will separate simple names from each other. Including all simple names and periods of the length of a name shall not exceed 44 characters. Thus, a maximum of 22 qualification levels will be possible for data set names.

b. Labeling Convention

(1) The OCDMS Software System shall adhere to the following conventions in order to ensure uniformity and noninterference of labels (symbolic addresses). A standardized label selection procedure shall be adopted. Labels may identify routines as well as areas of coding within a routine. They shall also be used to cross-reference a flowchart with a symbolic listing.

(2) The following conventions for labels shall be adhered to:

o A label will consist of two to eight characters in the order listed below:

L. AL. DDD
LL. L. DDD
LL. AL. DD

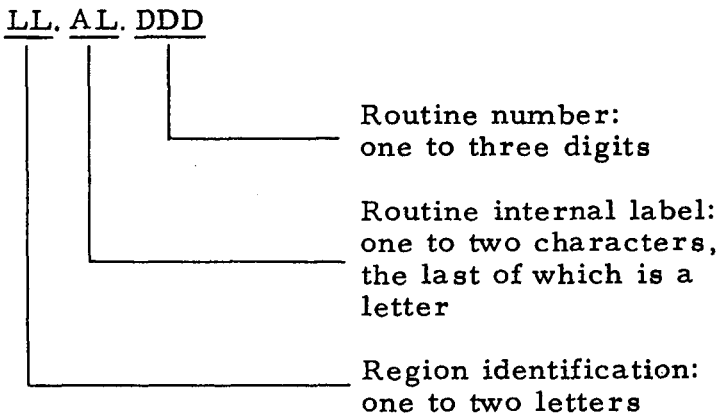
where

L = alphabetic letters (A-Z)

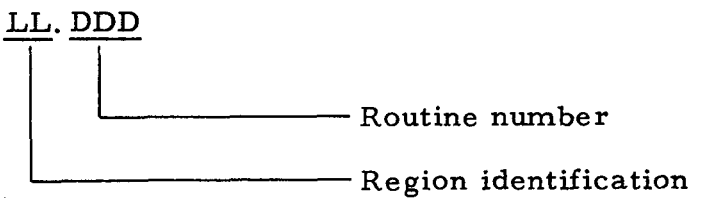
D = digits (0-9)

A = L or D

and



- A routine label consists only of the first and last character groups:



The combination of region identification and routine number shall uniquely identify the routines contained with a region

- Region identifiers shall be assigned a generic functional significance as the following table illustrates:

Category (1st Character)	Function
A	Symbol tables
C	Character attribute tables
G	Generators
M	Main programs
P	Printing routines
S	Source language input processor routines
T	Translator parsing routines

The subroutines which perform the task of processing source language input (S category) shall then be defined uniquely by the programmer as follows:

S3	Isolates next word from source statement
S15	Packs a character into a character position defined by a pointer address and a buffer address.

The remaining characters of the label shall be assigned any combination of letters or digits which the programmer might choose as long as "S3" (or "S15") are used consistently within the subroutine as the identifying characters of the label. For example, the symbols used in the subroutine labeling S15 might be as follows:

SA15	(Labeled instructions)
SB15	
SBB15	
SC15	
SD15	
SX15	Holds index register 1
SW15	Temporary storage
SV15	Variable storage

3.3.3 Program Modification

CPCEI modifications shall adhere to configuration management accounting practices delineated by Reference 2.3.1. The program development phase shall in general be conducted in accordance with guidelines and directives given by Reference 2.3.2.

3.3.4 CPCEI Testing Facilities

The OCDMS Support System CPCEI shall provide the techniques and services to ensure the ability to test all OCDMS computer program components during design, implementation, acceptance, and operations.

3.3.4.1 Test Services

System macro-instructions in conjunction with Support System control routines shall provide test services that include both action and control functions:

a. Test actions include the dumping (recording and display) of system tables, registers, and main storage, and the tracing of transfers, subroutine calls, and references to data.

b. Control capabilities shall include the dynamic testing of conditions resulting from program execution. Performance of test actions can accordingly be made independent on conditions detected during processing. When an error condition is detected, an attempt at recovery shall be made by alteration of application program data or control flow. Specific test actions shall be organized as subroutines or executed repetitively under loop control.

3.3.4.2 Test Procedures

Required job steps (i.e., assembly, program production, testing, edit of test data) shall be performed either as separate jobs or as steps within a single job.

The programmer shall have the capability to assemble part of his program independently.

The programmer shall be allowed the prerogative of testing a single aspect of the job by varying the input and selective editing of the output on a repetitive basis.

3.3.4.3 Test Operations

Test macro-instructions and application programs to be tested shall be assembled either together or separately. The assembler on

request shall produce a symbol table in addition to its normal output. This table shall contain with respect to application programs, the following items: (1) symbol names, (2) data attributes (type, length, and scale), and (3) named instruction attributes.

The symbol table shall be processed together with the assembly programs, macro-instructions, and control dictionaries. It shall be used during postprocessing to edit test data into a format that includes the symbolic names and data attributes of the source program.

3.3.5 CPCEI Expandability

The requirements specified in Sections 3.3.1 and 3.3.2 constitute a modular design concept relative to this CPCEI. Expandability to the CPCEI shall be provided by means of CPC modular construction techniques. Additions to the CPCEI shall conform to the requirements of Paragraph 3.3.3 with particular emphasis given to ensure that specified file protection, program protection, and program control conventions are not disregarded.

4.0 QUALITY ASSURANCE PROVISIONS

Requirements for formal verification of the performance of the CPCEI in accordance with the requirements for Section 3 of this specification are specified in order to:

a. Determine if the components of the CPCEI are implemented correctly

b. Determine if the CPCEI satisfies the requirements of its Part I Specification

c. Obtain test results that are used to determine if scheduled milestones have been achieved

d. Formally qualify the completed computer programs for operations use

The methods of verification that are specified herein include inspection of the CPCEI, review of analytical data, demonstration tests, and review of test data.

4.1 Implementation Test Requirements

Implementation tests include all tests of the CPCEI other than those accomplished during integration tests (see Paragraph 4.2). Several stages of tests shall be required to validate the design of the CPCEI and to verify that the implementation of the design is correct. These shall include, but not be limited to, the following categories:

a. Pre-Implementation Design Tests: Tests run on trial designs prior to establishing an initial design approach. These tests indicate real-time performance characteristics, computational accuracies, storage limitations, etc. Tests of this type shall, when appropriate, be continued throughout the design process.

b. Subprogram Checkout: Visual inspections and hand manipulations with selected data of coded CPCEI subprograms; followed by assembling the subprogram on the computer. Each assembled subprogram shall be tested by use of controlled data inputs. The goal is to identify and reduce indigenous and exogenous failure mechanisms prior to combining the subprograms into main programs or other functional program aggregates.

c. Main Program Checkout: Tests performed on functionally related subprograms. These shall be executed initially with no input to verify the ability to cycle. Controlled inputs shall

then be introduced to establish correct performance. Purpose of the testing is to eliminate logic and coding errors from the interfaces between subprograms. Testing levels shall be accomplished at this stage to the corresponding levels of subprograms within the CPCEI.

d. CPCEI Simulated Environment Tests

(1) The CPCEI shall be tested in a simulated environment prior to its integration into the total computer-based system. Such tests require the availability of a system environment simulator and associated test support tools.

(2) Simulated environment testing objectives are as follows:

- o To obtain a more controlled test of the CPCEI than could be accomplished in the total OCDMS
- o To determine the safety of the CPCEI in the OCDMS without exposing the system to unnecessary hazard
- o The OCDMS may not be available for use for preliminary qualification of the CPCEI prior to transferring it from a development facility to a using facility
- o To provide preliminary training in the use of the CPCEI and to evaluate proposed operating procedures for the CPCEI

4.1.1 Design and Development Testing

Computer program components and program tests shall be conducted in the acquisition phase prior to the preliminary qualification tests. These shall be validation and verification tests that prove the design and demonstrate specified performance requirements for each of the major functions.

a. Language Translation

(1) Translating OCDMS user-oriented language(s)

(2) Assembly processing

(3) Data formatting

(4) Data conversions

b. Data Management

(1) Data set organization

(2) Data access control methods

(3) Blocking and buffering facilities

c. Job Management

(1) Scheduling control

(2) Job file control

(3) Accounting processing

d. Task Management

(1) Single-task operations

(2) Multitask operations

(3) Storage allocation

(4) Task synchronization

(5) Exception condition handling

(6) Miscellaneous services

e. Program Production

(1) Module linkage

(2) Library input

(3) Program overlay

(4) Program modification

(5) Data base edit/update

(6) Object module conversion

f. Program Test and Verification

(1) Dumps

(2) Traces

(3) Tests

(4) Branches

(5) Initialize and set up

4.1.2 Preliminary Qualification Tests

Preliminary qualification tests shall verify each requirement of Section 3, which can be tested in a simulated environment. They will serve as the basis for transfer of the CPCEI to the NASA-MSFC facility.

4.1.2.1 Qualification Test Requirements

The procuring agency shall review procedures and audit results of critical demonstration tests which as a minimum shall include:

a. System compatibility: All computer program components (CPC's) shall be tested for proper program linkage and operation of the functional hardware units.

b. Man-machine relationships: Particular emphasis shall be given to qualifying the man-machine compatibility, and the adequacy of the man-machine to fulfill the mission requirements.

c. Simulated environment: Emphasis shall be given to simulating the most adverse conditions possible for the computer and other OCDMS hardware elements during demonstration of the software system.

4.1.2.2 Resources Required for Testing (to be determined)

4.1.2.3 Test Schedules and Locations (to be determined)

4.1.3 Special Test Requirements (to be determined)

4.2 Integration Test Requirements

This section specifies the verification test requirements applicable to performance/design requirements identified in Section 3.0, which cannot be accomplished until the CPCEI is assembled into or used with the OCDMS computer-based system environment and other CPCEI's.

4.2.1 General

The OCDMS Support System CPCEI tests that are required in direct support of system integration are as follows:

4.2.1.1 Sequence of Tests (to be determined)

4.2.1.2 Functions To Be Tested (to be determined)

4.2.1.3 Testing Environment (to be determined)

4.2.1.4 Support Computer Programs Required (to be determined)

4.2.1.5 Personnel Required (to be determined)

4.2.1.6 Equipment Required (to be determined)

4.2.2 Acceptance/Qualification Test

The requirements imposed against the CPCEI for formal qualification of the integrated computer program components (CPC's) with OCDMS are identified in the following subparagraphs. Verification of the requirements shall be accomplished by inspection or review of analytical data, or by demonstration, or test and review of test data, or a combination of these as required by the procuring agency.

4.2.2.1 Sequence of Tests (to be determined)

4.2.2.2 Functions To Be Tested (to be determined)

4.2.2.3 Testing Environment (to be determined)

4.2.2.4 Support Computer Programs Required (to be determined)

4.2.2.5 Personnel Required (to be determined)

4.2.2.6 Equipment Required (to be determined)

6.0 NOTES

None.

10.0 APPENDIX

10.1 Bose-Chaudhuri Error Protection Encoding Scheme

Apollo Command Program and OCDMS shall use the Bose-Chaudhuri error protection encoding scheme. Using this code, the limits on the probability of an undetected error can be stated independently of transmission line error statistics. The mathematical technique for developing this code is described below and is followed by a programming interpretation.

Four program constraints which must be determined premission to be incorporated into the command program are defined as follows:

a. n = total number of information bits plus error protection bits

b. k = total number of information bits

c. p = total number of error protection bits (p = n - k)

d. G = an operator function (generator polynomial) consisting of a fixed format of p + 1 binary bits

The mathematical technique is as follows. The error bits can be shown to be a portion of a polynomial that is derived by dividing a polynomial, F(x), made up of n terms, by G(x). The coefficients of the information bits are used in F(x). This will yield a Q(x) and a remainder R(x):

$$\frac{F(x)}{G(x)} = Q(x) + \frac{R(x)}{G(x)}$$

Using exclusive OR addition, F(x) is added to R(x) to yield:

$$F(x) - R(x)$$

The coefficients of the first k bits are the information bits. The remaining bits are the required error protection bits. F(x) - R(x) is the data received at the remote site. This data should be exactly divisible by Q(x). If a remainder other than zero is obtained after dividing F(x) - R(x) by Q(x), an error has occurred.

The general form of the polynomials discussed above will be as follows:

$$F(x) = A_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_p x^p$$

$$+ a_{p-1}x^{p-1} + \ldots + a_1 x^1 + a_0 x^0$$

$$G(x) = b_p x^p + b_{p-1}x^{p-1} + \ldots + b_1 x^1 b_0 x^0$$

It is desired to transmit the information bits 10101 plus three error protection bits. Thus, $n = 8$, $k = 5$, and $p = 3$. Form $F(x)$:

$$F(x) = 1 \cdot x^7 + 0 \cdot x^6 + 1 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 0 \cdot x^0$$

$$F(x) = x^7 + 0 + x^5 + 0 + x^3 + 0 + 0 + 0$$

The constant $G(x)$ is defined as 1011. The following polynomial is formed:

$$G(x) = 1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$$

$$G(x) = x^3 + 0 + x + 1$$

Using exclusive OR addition instead of subtracting, divide $F(x)$ by $G(x)$:

$$\begin{array}{r|l} & x^4 + x + 1 \\ \hline x^3 + 0 + x + 1 & x^7 + 0 + x^5 + 0 + x^3 + 0 + 0 + 0 \\ & \underline{x^7 + 0 + x^5 + x^4} \\ & \quad x^4 + x^3 + 0 + 0 \\ & \quad \underline{x^4 + 0 + x^2 + x} \\ & \qquad x^3 + x^2 + x + 0 \\ & \qquad \underline{x^3 + 0 + x + 1} \end{array}$$

This yields $Q(x) = x^4 + x + 1$ and $R(x) = x^2 = 1$.

Form the polynomial $F(x) - R(x)$, using exclusive OR addition:

$$\begin{array}{r} x^7 + 0 + x^5 + 0 + x^3 + 0 + 0 + 0 \\ \hline 1 \quad + 0 + 1 \quad + 0 + 1 \quad + 1 + 0 + 1 \end{array} \qquad \begin{array}{l} \\ x^2 + 0 + 1 \end{array}$$

Therefore, the error protection bits are 101 and the entire message is 10101101 . The OCDMS supervisory routines will receive $F(x) - R(x)$ and will divide it by $G(x)$. This will yield a remainder of zero if the information is error free.

Constants used in the system are as follows:

a. 600-bit block

n = 600

k = 567

p = 33

b. 60-bit message subblocks

n = 57

k = 30

p = 27

c. 40-bit return subblocks

n = 40

k = 16

p = 14

10.2 Glossary

Access method: Any of the data management techniques available to the user for transferring data between main storage and an input/output device.

Address constant: A value, or an expression representing a value, used in the calculation of storage addresses.

Allocate: To grant a resource to, or reserve it for, a job or task.

Application program: Any of the class of routines that perform processing of experiment procedures for checkout, data management, equipment self-check, etc.

ASCII code: American standard code for information interchange.

Asynchronous: Without regular time relationship; hence, as applied to program execution, unexpected or unpredictable with respect to instruction sequence.

Attach (task): To create a task control block and present it to the supervisor control program.

Attribute: A characteristic (e.g., attributes of data include record length, record format, data set name, associated device type and volume identification, use, creation date).

Auxiliary storage: Data storage other than main storage.

Binary: A characteristic, property, or condition in which there are two possible alternatives. The binary number system uses two as its base.

BCD (binary-coded decimal): A number usually consisting of successive groups of figures, in which each group of four figures is a binary number that represents but does not necessarily equal arithmetically, a particular figure in an associated decimal number.

Block (records): (1) To group records for the purpose of conserving storage space or increasing the efficiency of access or processing. (2) A physical record so constituted, or a portion of a telecommunications message defined to be a unit of data transmission.

Block loading: The form of fetch that brings control sections of a load module into contiguous positions of main storage.

Buffer (program input/output): A portion of main storage into which data is read, or from which it is written.

Checkpoint: (1) A point at which information about the status of a job step can be recorded so that the job step can be restarted. (2) To record such information.

Control block: A storage area through which a particular type of information required for control of the operating system is communicated among its parts.

Control dictionary: The external symbol dictionary and relocation dictionary, collectively, of an object or load module.

Control program: A collective or general term for all routines in the operating system that contribute to the management of resources, implement the data organization or communications conventions of the operating system, or contain privileged operations.

Control section: The smallest separately relocatable unit of a program; that portion of text specified by the programmer to be an entity, all elements of which are to be loaded into contiguous main storage locations.

Data control block: A control block through which the information required by access routines to store and retrive data is communicated to them.

Data definition name (dd name): A name appearing in the data control block of a program which corresponds to the name field of a data definition statement.

Data definition (DD) statement: A job control statement that describes a data set associated with a particular job step.

Data management: A general term that collectively describes those functions of the control program that provide access to data sets, enforce data storage conventions, and regulate the use of input/output devices.

Data organization: A term that refers to any one of the data management conventions for the arrangement of a data set.

Data set: The major unit of data storage and retrieval in the operating system, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

Data set control block (DSCB): A data set label for a data set in direct-access storage.

Data set label (DSL): A collection of information that describes the attributes of a data set, and that is normally stored with the data set; a general term for data set control blocks and tape data set labels.

Decimal: A system of reckoning by 10 or power of 10 using digits 0-9 to express numerical quantities.

Deferred exit: The passing of control to a subroutine at a time determined by an asynchronous event rather than at a predictable time.

Dispatching priority: A number assigned to tasks, and used to determine precedence for use of the central processing unit in a multitask situation.

Dump (main storage): (1) To copy the contents of all or part of main storage onto an output device, so that it can be examined. (2) The data resulting from (1). (3) A routine that will accomplish (1).

EBCDIC: Extended BCD interchange code.

Entry point: Any location in a program to which control can be passed by another program.

Event: An occurrence of significance to a task; typically, the completion of an asynchronous operation, such as input/output.

Event control block (ECB): A control block used to represent the status of an event.

Exchange buffering: A technique using data chaining for eliminating the need to move data in main storage, in which control of buffer segments and user program work areas is passed between data management and the user program according to the requirements for work areas, input buffers, and output buffers, on the basis of their availability.

Exclusive segments: Segments in the same region of an overlay program, neither of which is in the path of the other. They cannot be in main storage simultaneously.

Execute (EXEC) statement: A job control statement that designates a job step by identifying the load module to be fetched and executed.

External reference: A reference to a symbol defined in another module.

External symbol: A control section name, entry point name, or external reference; a symbol contained in the external symbol dictionary.

Fetch (program): (1) To obtain requested load modules and transfer them into main storage, relocating them as necessary. (2) A control routine that accomplishes (1).

Floating point (floating-point arithmetic): A method of calculation which automatically accounts for the location of radix point. This is usually accomplished by handling the number as a signed mantissa times the radix raised to an integral exponent.

Generation data group: A collection of successive historically related data sets.

Hexadecimal: A number usually of more than one figure representing a sum in which the quantity represented by each figure is based on a radix of 16.

Initial program loading (IPL): As applied to the support system, the initialization procedure that loads the nucleus and begins normal operations.

Initiator/terminator: The job scheduler function that selects jobs and job steps to be executed, allocates input/output devices for them, places them under task control, and at completion of the job, supplies control information for writing job output on a system output unit.

Input job stream: A sequence of job control statements entering the the system, which may also include input data.

Input work queue: A queue of summary information of job control statements maintained by the job scheduler, from which it selects the jobs and job steps to be processed.

Installation: A general term for a particular computing system, in the context of the overall function it serves and the individuals who manage it, operate it, apply it to problems, service it, and use the results it produces.

Job: An externally specified unit of work for the computing system from the standpoint of installation accounting and operating system control. A job consists of one or more job steps.

Job control statement: Any one of the control statements in the input job stream that identifies a job or defines its requirements.

Job management: A general term that collectively describes the functions of the job scheduler and master scheduler.

Job scheduler: The control program function that controls input job streams and system output, obtains input/output resources for jobs and job steps, attaches tasks corresponding to job steps, and otherwise regulates the use of the computing system by jobs.

Job (JOB) statement: The control statement in the input job stream that identifies the beginning of a series of job control statements for a single job.

Job step: A unit of work for the computing system from the standpoint of the user, presented to the control program by job control statements as a request for execution of an explicitly identified program and a description of resources required by it. A job step consists of the external specifications for work that is to be done as a task or set of tasks. Hence, also used to denote the set of all tasks which have their origin in a job step specification.

Language translator: A general term for any assembler, compiler, or other routine that accepts statements in one language and produces equivalent statements in another language.

Library: In general, a collection of objects (e.g., data sets, volumes, card decks) associated with a particular use, and the location of which is identified in a directory of some type. In this context, see job library, link library, system library.

Linkage: The means by which communication is effected between two routines or modules.

Load: To fetch (i.e., to read a load module into main storage preparatory to executing it).

Load module: The output of the program production programs in a format suitable for loading into main storage for execution.

Locate mode: A transmittal mode in which data is pointed to rather than moved.

Logical record: A record from the standpoint of its content, function, and use rather than its physical attributes, (i.e., one that is defined in terms of the information it contains).

Macro-instruction: A general term used to collectively describe a macro-instruction statement, the corresponding macro-instruction definition, the resulting assembler language statements, and the machine language instructions and other data produced from the assembler language statements; loosely, any one of these representations of a machine language instruction sequence.

Main storage: All addressable storage from which instructions can be executed or from which data can be loaded directly into registers.

Master scheduler: The control program function that responds to operator commands, initiates actions requested thereby, and returns requested or required information; thus, the overriding medium for controlling the use of the computing system.

Module (programming): The input to, or output from, a single execution of an assembler, compiler, or linkage editor; a source, object, or load module; hence, a program unit that is discreet and identifiable with respect to compiling, combining with other units, and loading.

Move mode: A transmittal mode in which data is moved between the buffer and the user's work area.

Multijob operation: A term that describes concurrent execution of job steps from two or more jobs.

Multiprogramming: A general term that expresses use of the computing system to fulfill two or more different requirements concurrently.

Multitask operation: Multiprogramming; called multitask operation to express parallel processing not only of many programs, but also of a single reenterable program used by many tasks.

Name: A set of one or more characters that identifies a statement, data set, module, etc., and that is usually associated with the location of that which it identifies.

Nucleus: That portion of the control program that must always be present in main storage. Also, the main storage area used by the nucleus and other transient control program routines.

Object module: The output of a single execution of an assembler or compiler, which constitutes input to program production. An object module consists of one or more control sections in relocatable, though not executable, form and an associated control dictionary.

Octal: Pertaining to eight; usually describing a number system at base eight.

Operator command: A statement to the control program, issued via the console device, which causes the control program to provide requested information, alter normal operations, initiate new operations, or terminate existing operations.

Output writer: A job scheduler function that transcribes specified output data sets onto a system output unit, independently of the program that produced such data sets.

Output work queue: A queue of control information describing system output data sets, which specifies to an output writer the location and disposition of system output.

Overlay: To place a load module or a segment of a load module into main storage locations occupied by another load module or segment.

Overlay (load) module: A load module that has been divided into overlay segments, and has been provided by program production with information that enables overlay supervisor to implement the desired loading of segments when requested.

Overlay supervisor: A control routine that initiates and controls fetching of overlay segments on the basis of information recorded in the overlay module by program production.

Path: A series of segments which, as represented in an overlay tree, form the shortest distance in a region between a given segment and the root segment.

Physical record: A record from the standpoint of the manner or form in which it is stored, retrieved, and moved, (i.e., one that is defined in terms of physical qualities).

Post: To note the occurrence of an event.

Priority scheduling system: A form of job scheduler which uses input and output work queues to improve system performance.

Qualified name: A data set name that is composed of multiple names separated by periods.

Qualifier: All component names in a qualified name other than the rightmost (which is called the simple name).

Reader/interpreter: A job scheduler function that services an input job stream.

Ready condition: The condition of a task such that it is in contention for the central processing unit, all other requirements for its activation having been satisfied.

Record: A general term for any unit of data that is distinct from all others when considered in a particular context.

Reenterable: The attribute of a load module that allows the same copy of the load module to be used concurrently by two or more tasks.

Region: A contiguous area of main storage within which segments can be loaded independently of paths in other regions. Only one path within a region can be in main storage at one time.

Relocation: The modification of address constants required to compensate for a change of origin of a module or control section.

Relocation dictionary: That part of an object or load module which identifies all relocatable address constants in the module.

Resource: Any facility of the computing system or operating system required by a job or task and including main storage, input/output devices, the central processing unit, data sets, and control and processing programs.

Restart: To reestablish the status of a job using the information recorded at a checkpoint.

Return code: A value that is by system convention placed in a designated register (the "return code register") at the completion of a program. The value of the code, which is established by user convention, may be used to influence the execution of succeeding programs or, in the case of an abnormal end of task, it may simply be printed for programmer analysis.

Return code register: A register identified by system convention in which a user-specified condition code is placed, at the completion of a program.

Reusable: The attribute of a routine that enables the same copy of the routine to be used by two or more tasks. (See reenterable, serially reusable.)

Scheduler: (See master scheduler and job scheduler.)

Secondary storage: Auxiliary storage.

Serially reusable: The attribute of a routine that when in main storage the same copy of the routine can be used by another task after the current use has been concluded.

Service program: Any of the class of standard routines that assist in the use of a computing system and in the successful execution of problem programs, without contributing directly to control of the system or production of results, and including utilities, simulators, and test and debugging routines.

Simple buffering: A technique for controlling buffers in such a way that the buffers are assigned to a single data control block and remain so assigned until the data control block is closed.

Source module: A series of statements in the symbolic language of an assembler or compiler, which constitutes the entire input to a single execution of the assembler or compiler.

Stacked job processing: A technique that permits multiple job definitions to be grouped (stacked) for presentation to the system, which automatically recognizes the jobs, one after the other. More advanced systems allow job definitions to be added to the group (stack) at any time and from any source, and also honor priorities.

Substitute mode: A transmittal mode used with exchange buffering in which segments are pointed to and exchanged with user work areas.

Subtask: A task that is created by another task by means of the ATTACH macro-instruction.

Supervisor: A routine or routines executed in response to a requirement for altering or interrupting the flow of operations through the central processing unit, or for performance of input/output operations, and therefore, the medium through which the use of resources is coordinated and the flow of operations through the central processing unit is maintained; hence, a control routine that is executed in supervisor state.

Synchronous: Occurring concurrently, and with a regular or predictable time relationship.

System input unit. A device specified as a source of an input job stream.

System library: The collection of all cataloged data sets at an installation.

System macro-instruction: A predefined macro-instruction that provides access to operating system facilities.

System output unit: An output device shared by all jobs, onto which specified output data is transcribed.

Task: A unit of work for the central processing unit from the standpoint of the control program; therefore, the basic multiprogramming unit under the control program.

Task control block (TCB): The consolidation of control information related to a task.

Task dispatcher: The control program function that selects from the task queue the task that is to have control of the central processing unit, and gives control to the task.

Task management: A general term that collectively describes those functions of the control program that regulate the use by tasks of the central processing unit and other resources (except for input/output devices).

Task queue: A queue of all the task control blocks present in the system at any one time.

Test translator: A facility that allows various debugging procedures to be specified in assembler language programs.

Text: The control sections of an object or load module, collectively.

Throughput: A measure of system efficiency; the rate at which work can be handled by a computing system.

Transmittal mode: The method by which the contents of an input buffer are made available to the program, and the method by which a program makes records available for output.

Turnaround time: The elapsed time between submission of a job to a computing center and the return of results.

Wait condition: As applied to tasks, the condition of a task such that it is dependent on an event or events in order to enter the ready condition.